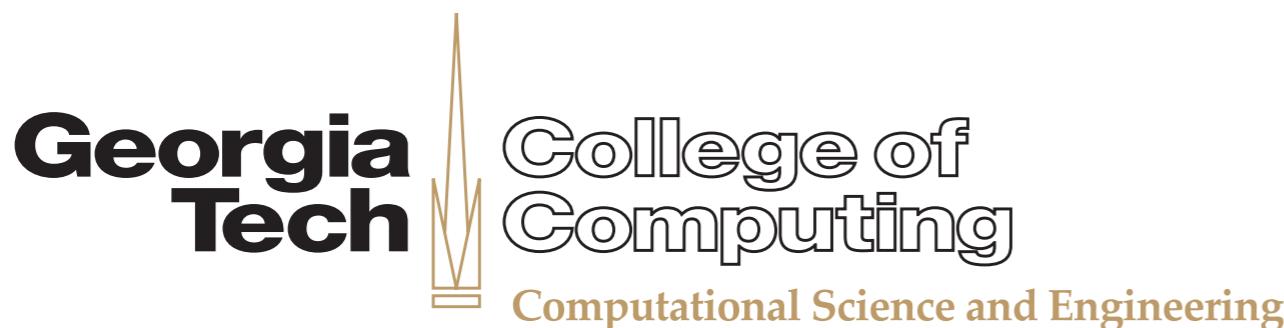
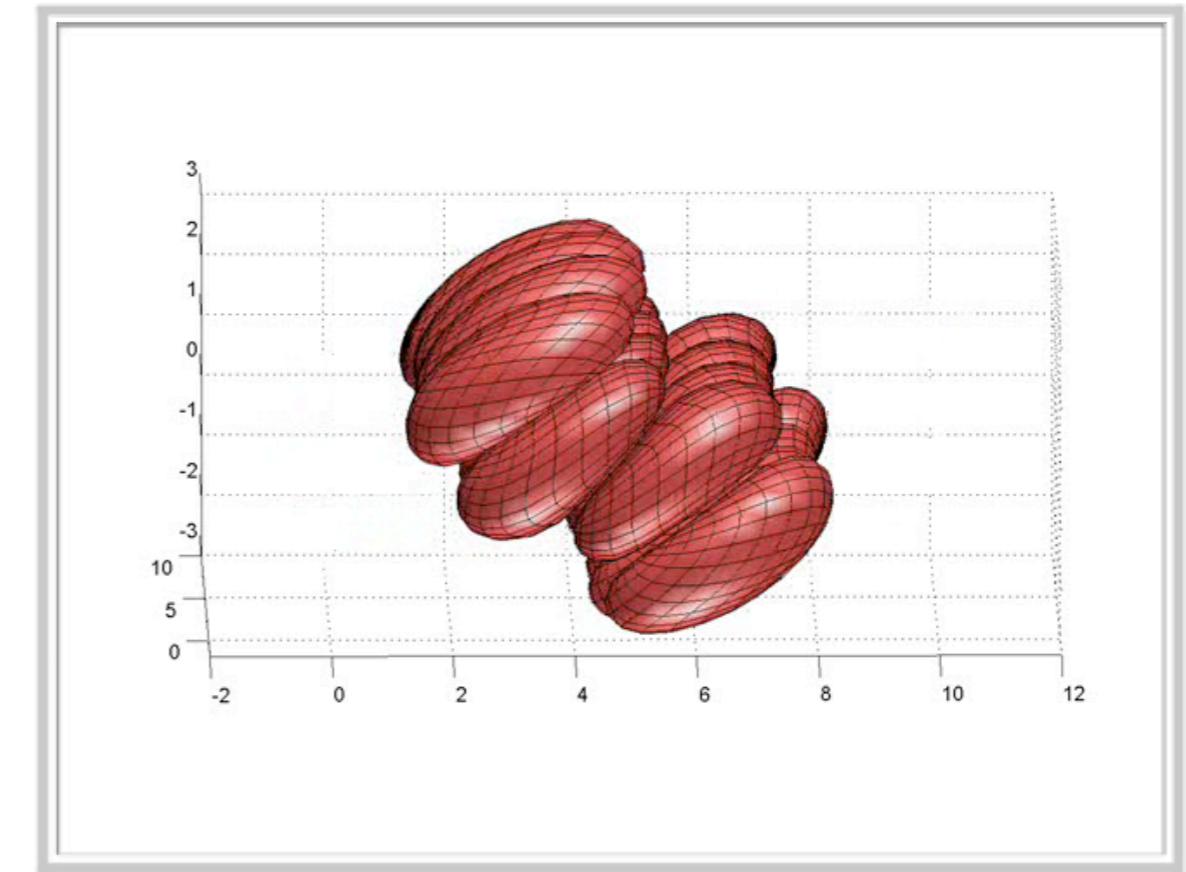
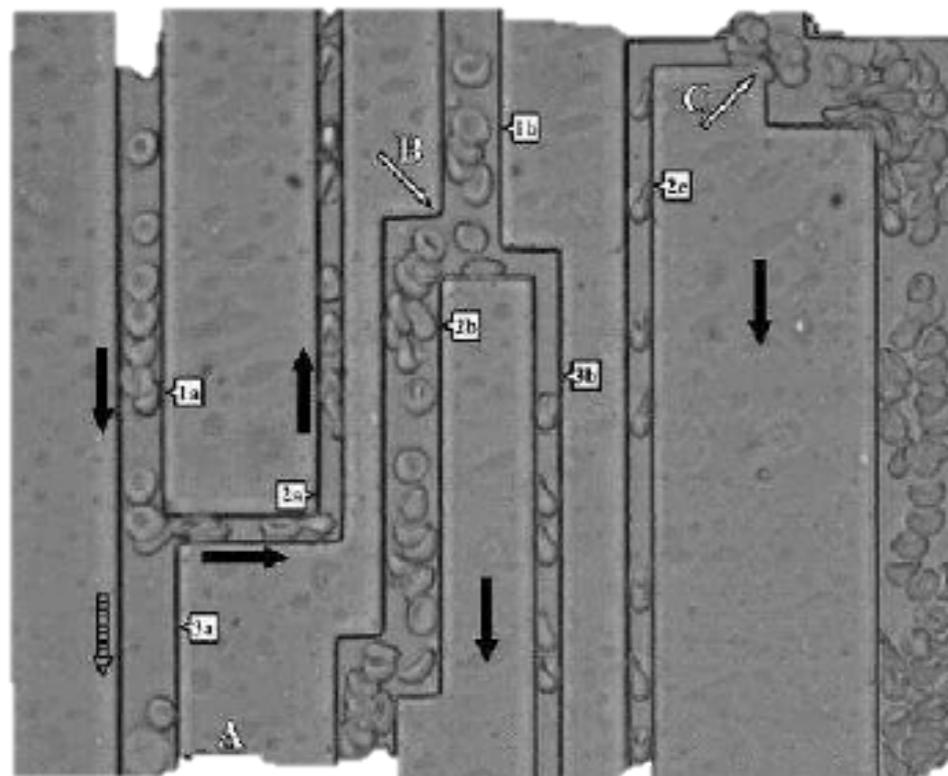
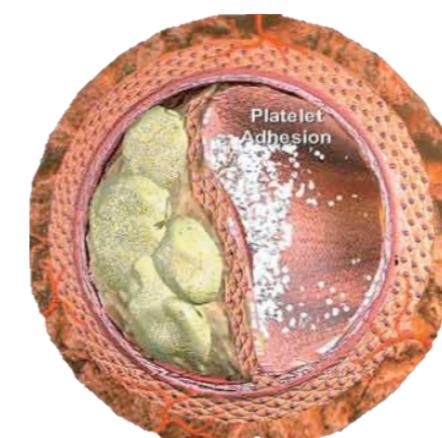
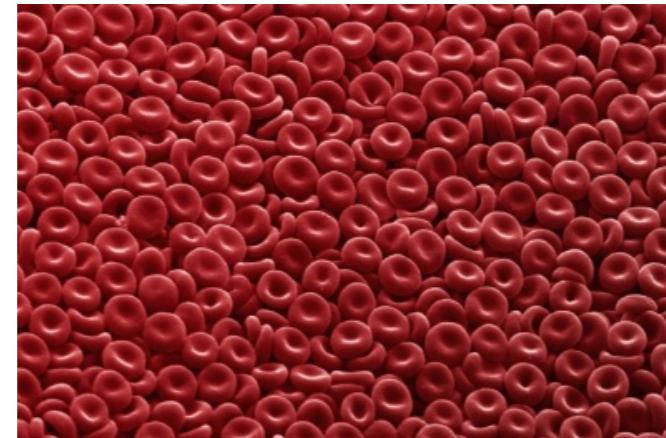
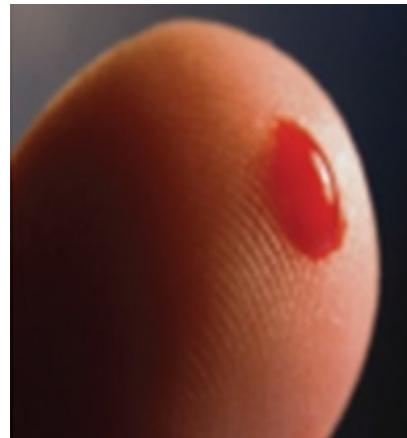


PERFORMANCE PRINCIPLES & PRACTICE

Casey Battaglino · Aparna Chandramowlishwaran · Jee Choi
· Kent Czechowski · Cong Hou · Chris McClanahan · David S. Noble, Jr.
· Richard (Rich) Vuduc

Salishan Conference on High-Speed Computing
Gleneden Beach, Oregon — April 27, 2011





CONTEXT: MoBo ("MOVING BOUNDARIES")

Citation: A. Rahimian, I. Lashuk, A. Chandramowlishwaran, D. Malhotra, L. Moon, R. Sampath, A. Shringarpure, S. Veerapaneni, J. Vetter, R. Vuduc, D. Zorin, and G. Biros. "Petascale direct numerical simulation of blood flow on 200k cores and heterogeneous architectures." In SC'10. Winner, Gordon Bell Prize. <http://dx.doi.org/10.1109/SC.2010.42>

- ◉ **Hardware/software co-design**

vs.

- ◉ **Algorithm-architecture co-design**

- Hardware
- vs.
- Algorithm



PASCAL'S WAGER

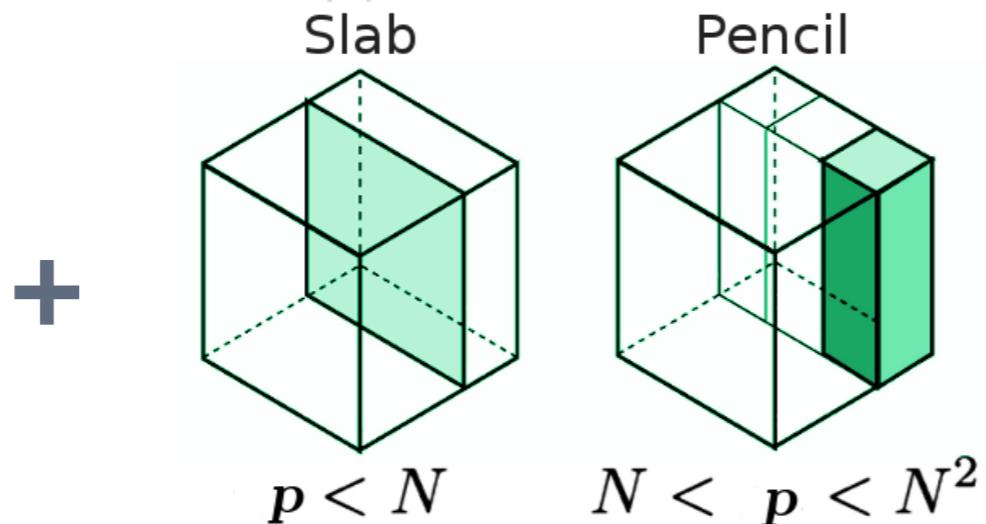
Because reducing eternity
to a crapshoot is so inspiring.



vs.



Swim lane 1



Swim lane 2

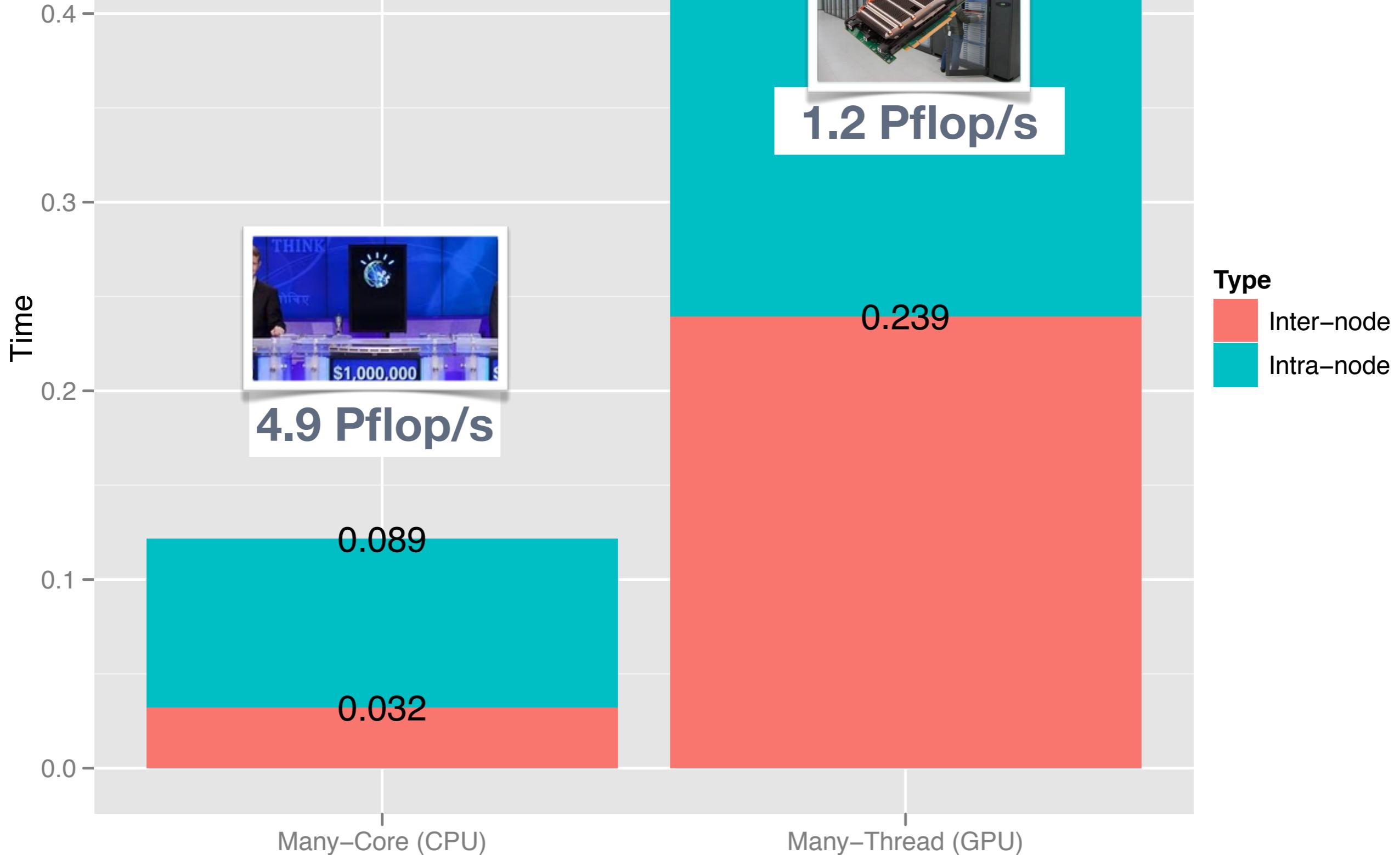
= ?

3D FFT

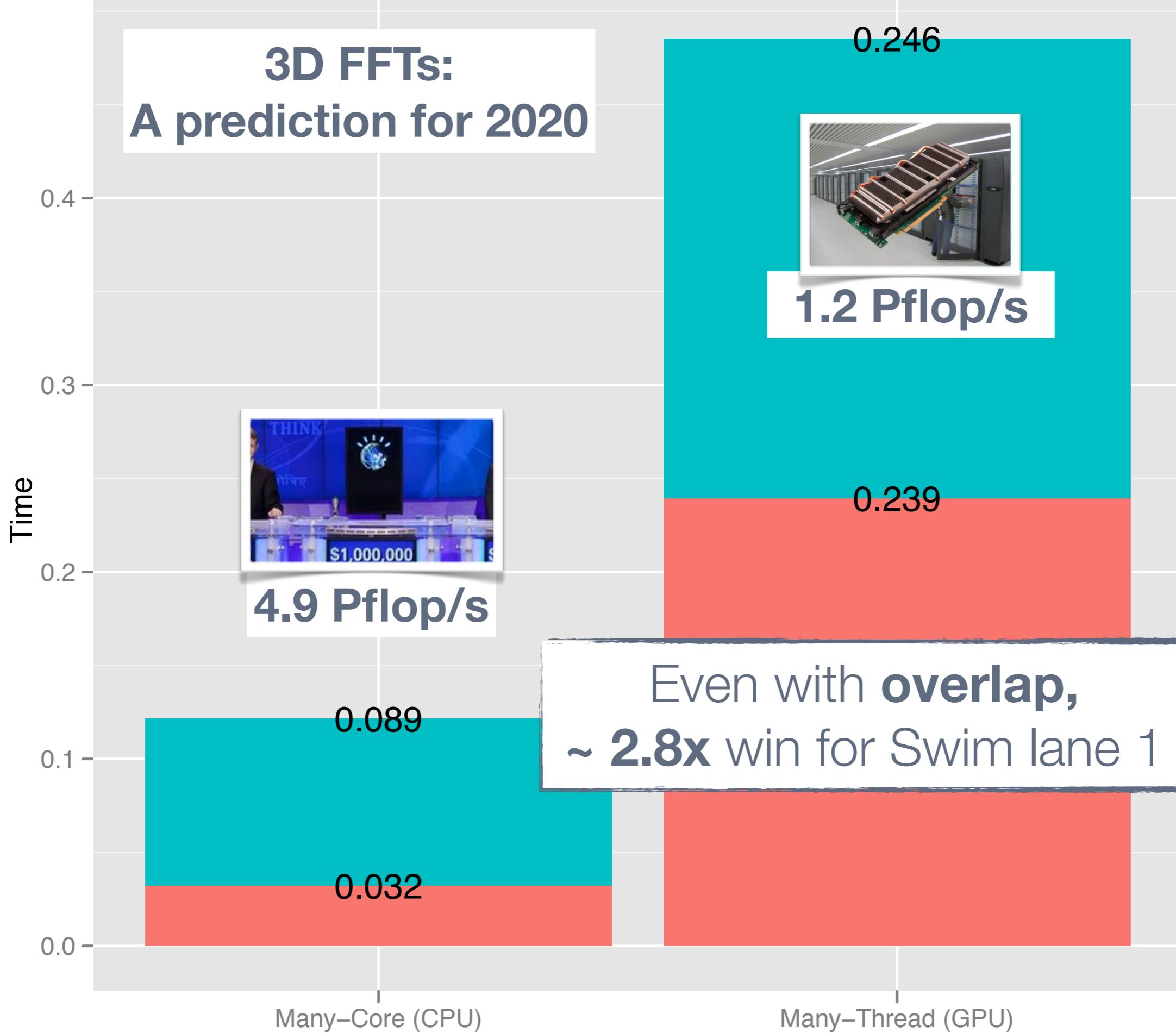
<http://socialmediab2b.com/wp-content/uploads/2011/02/IBM-Watson-Jeopardy.jpg>

<http://legitreviews.com/images/news/2010/Tianhe-1A.jpg>

3D FFTs: A prediction for 2020



3D FFTs: A prediction for 2020



Question: How do principles inform practice?

Question: How do principles inform practice?

Answers (?):

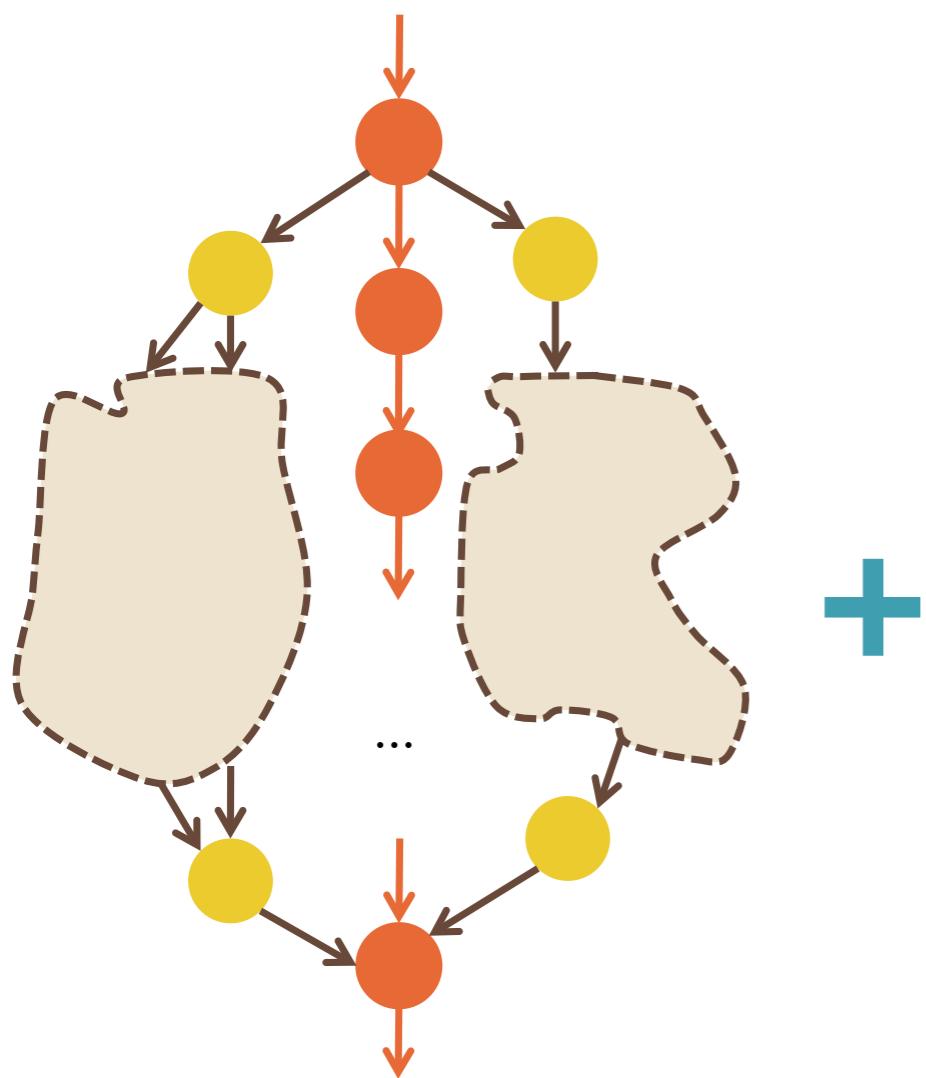
- 1.** Make algorithm-architecture co-design precise
- 2.** Suggest performance analysis metrics
- 3.** Imply a programming style

Question: How do principles inform practice?

Answers (?):

- 1. Make algorithm-architecture co-design precise**
- 2. Suggest performance analysis metrics**
- 3. Imply a programming style**

K. Czechowski, C. Battaglino, C. McClanahan, A. Chandramowliswaran, R. Vuduc. “Balance principles for architecture-algorithm co-design.” In *HotPar’11*.

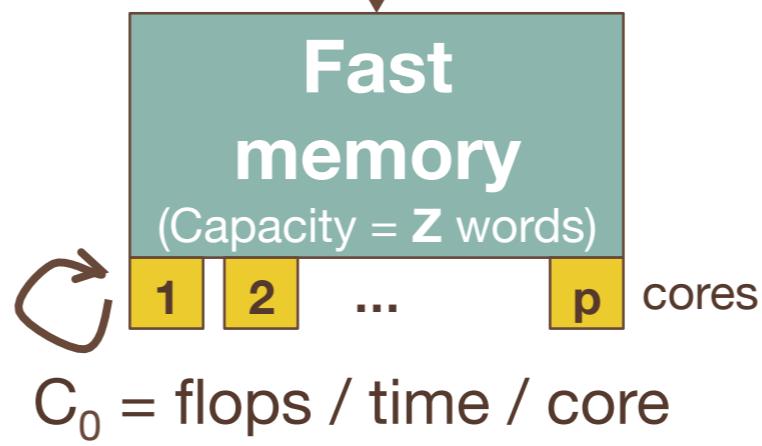


+

Slow memory

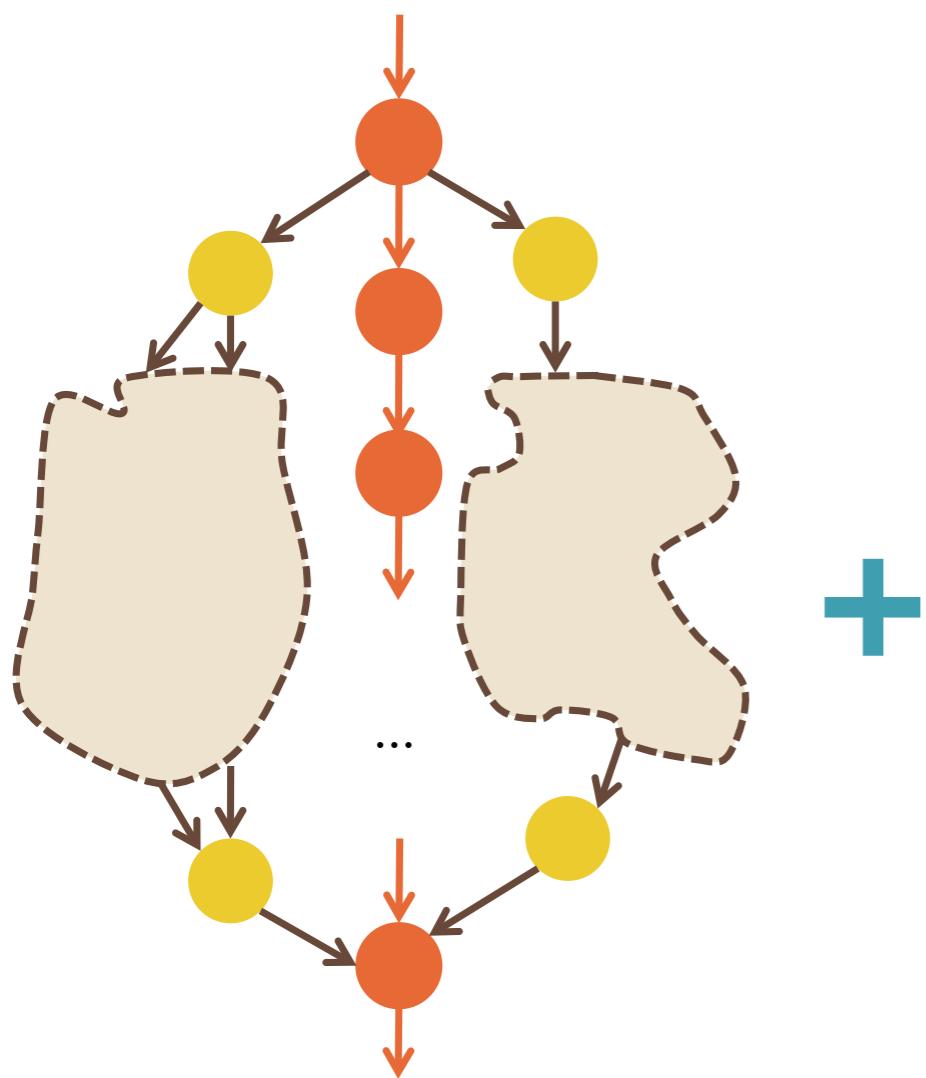
Transaction size
= L words

α = latency
 β = bandwidth



= ?

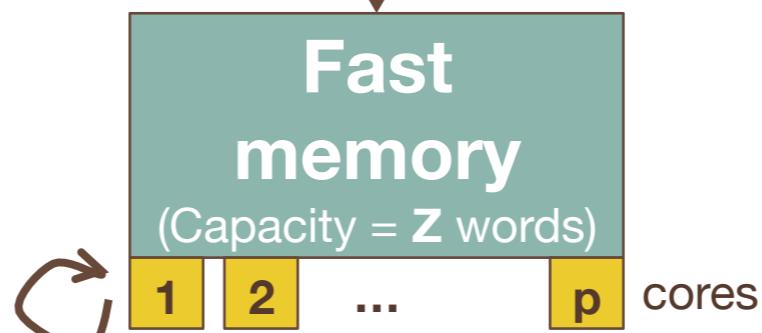
First principles



Slow memory

Transaction size
= L words

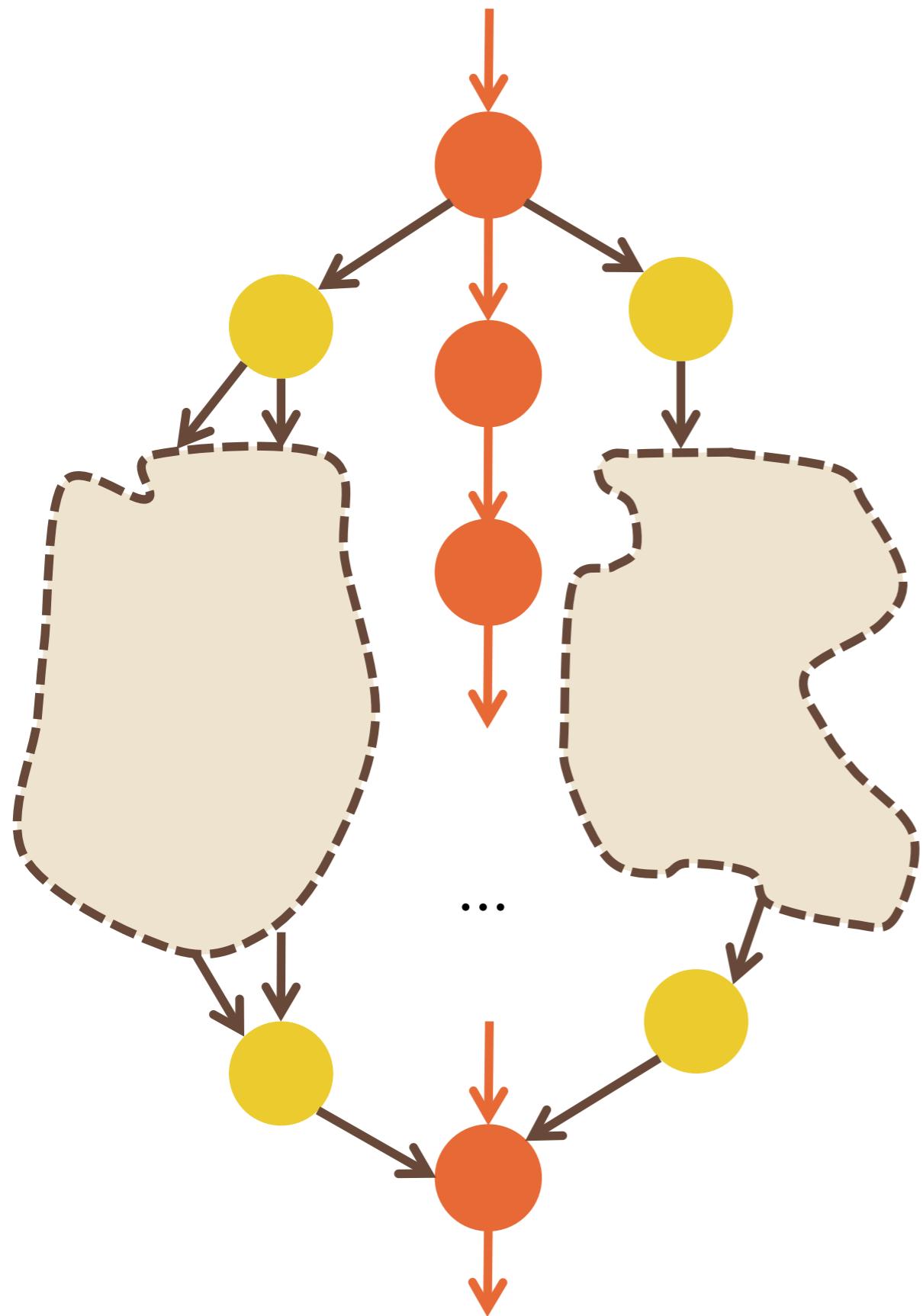
α = latency
 β = bandwidth



$$C_0 = \text{flops} / \text{time} / \text{core}$$

= ?

1. Design and analyze an **efficient parallel** algorithm.
2. Design and analyze an **I/O-efficient** algorithm.
3. Mix well.
4. Cook-up an architectural **cost model**.



$W(n)$ = *work* (total ops)

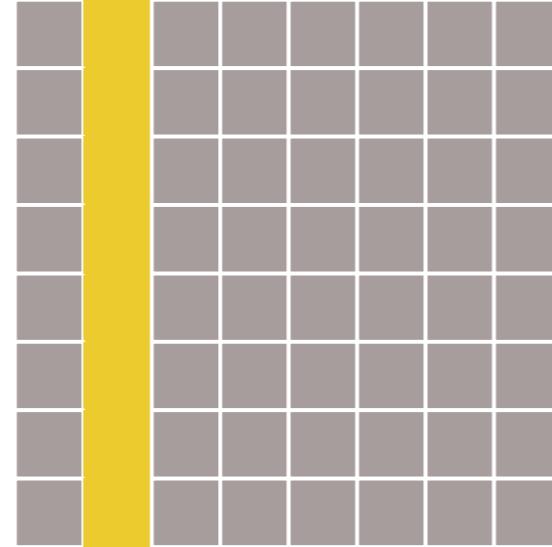
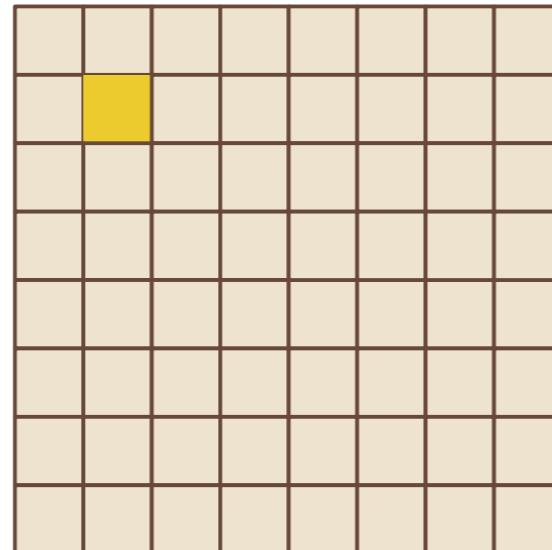
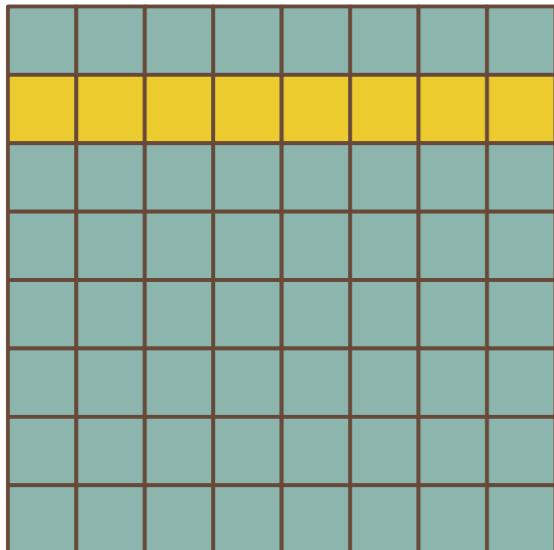
$D(n)$ = *depth* (critical path)

$W(n) / D(n)$
= inherent parallelism

Desiderata:

- Work optimality
- Maximal parallelism

Example: Matrix multiply (non-Strassen)

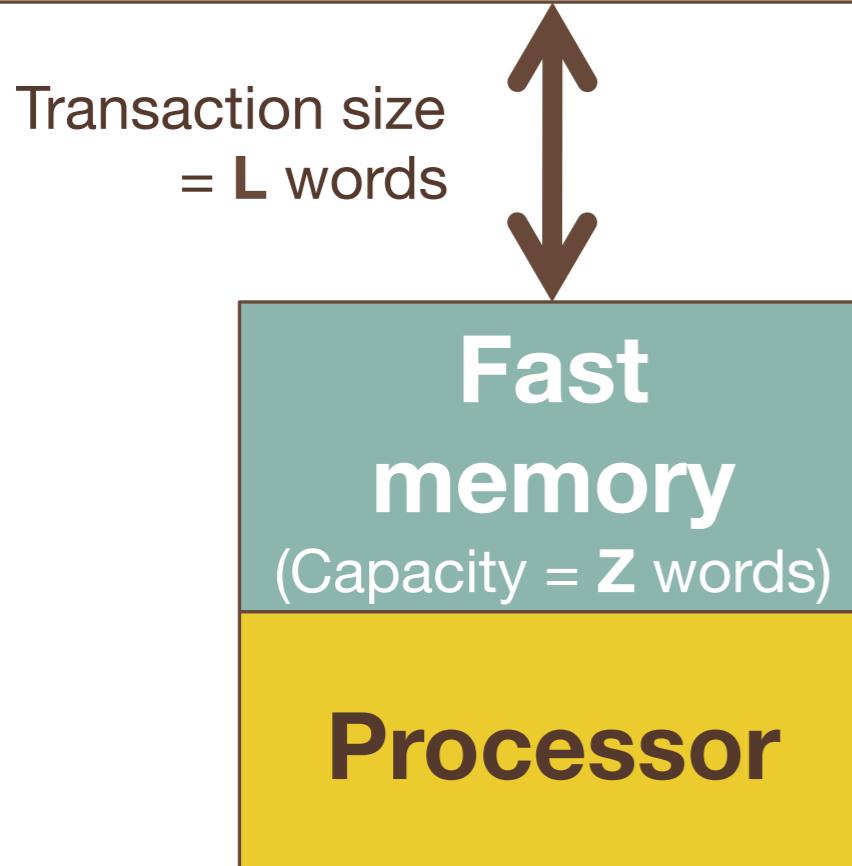


C \leftarrow **C** + **A** * **B**

$$\frac{W(n)}{D(n)} = \Omega\left(\frac{n^3}{\log n}\right)$$

Parallelism

Slow memory



$W(n)$ = work (total ops)

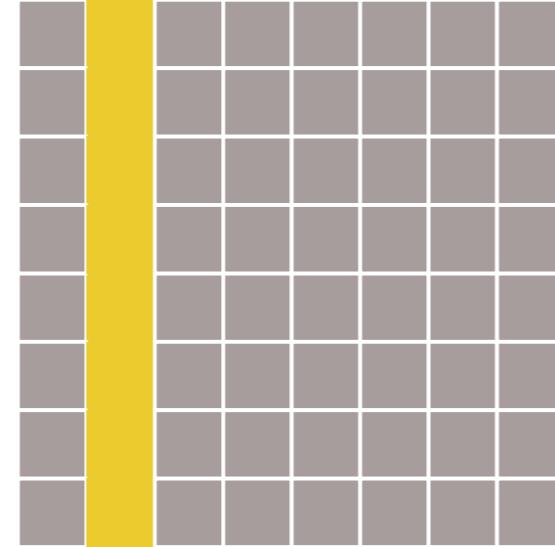
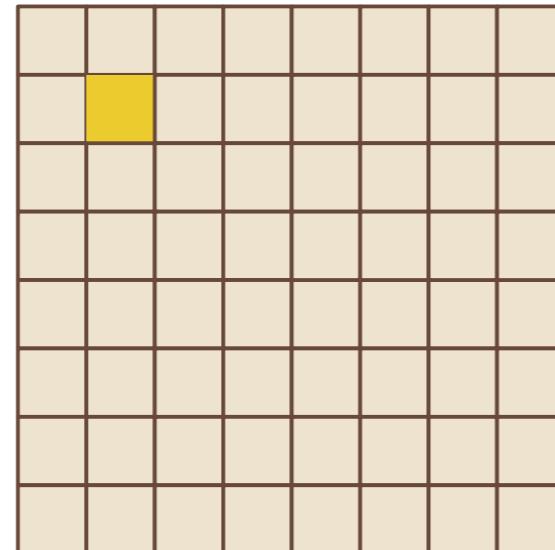
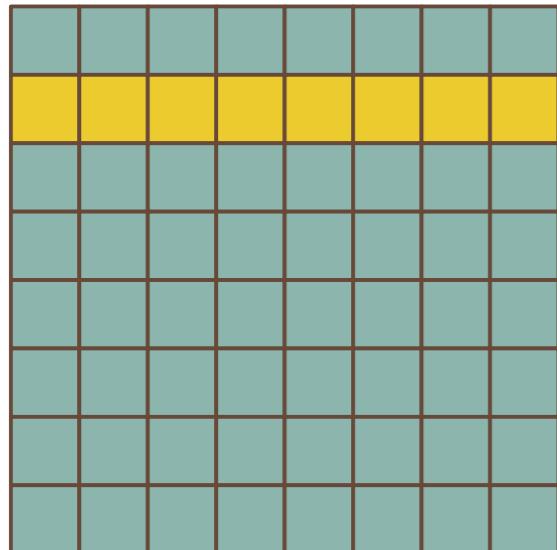
$Q(n; Z, L)$ = no. transfers

$W(n) / (Q(n; Z, L) * L)$
= computational intensity
(ops : words)

Desiderata:

- Work optimality
- Maximal intensity

Example: Matrix multiply (non-Strassen)

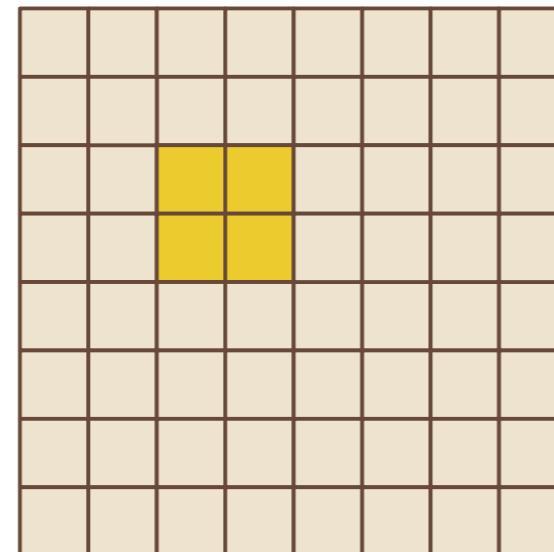
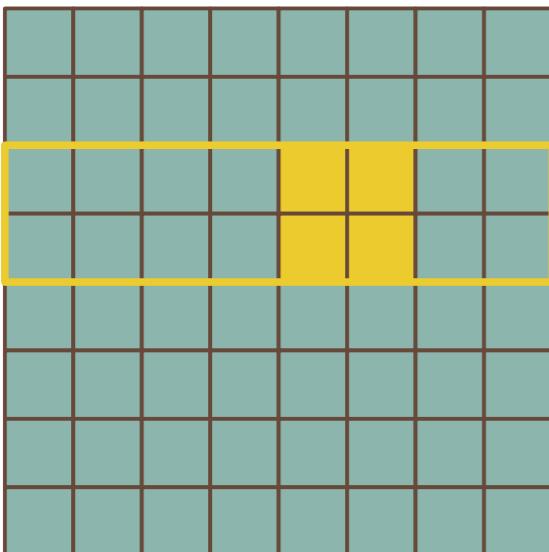
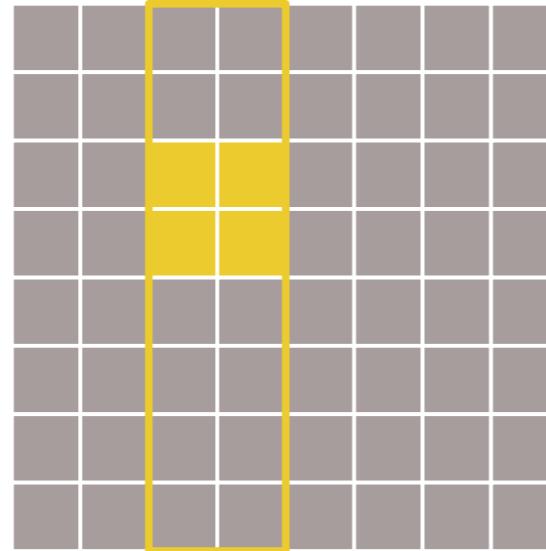


C \leftarrow **C** + **A** * **B**

$$\frac{W(n)}{Q(n; Z, L) \cdot L} = \Theta(1)$$

Intensity

Example: Matrix multiply (non-Strassen)



$\boxed{\mathbf{C}} \leftarrow \boxed{\mathbf{C}} + \boxed{\mathbf{A}} * \boxed{\mathbf{B}}$

$$Q(n; Z, L) = \Omega\left(\frac{n^3}{L\sqrt{Z}}\right)$$

Assumes **contiguous layout**.
Result is **optimal**.

$$\frac{W}{Q \cdot L} = \mathcal{O}\left(\sqrt{Z}\right)$$

Intensity

So far, we've considered parallelism
and I/O separately.

Combining them requires
scheduling and memory hierarchy
assumptions.

Example: Work-stealing + core-private caches.

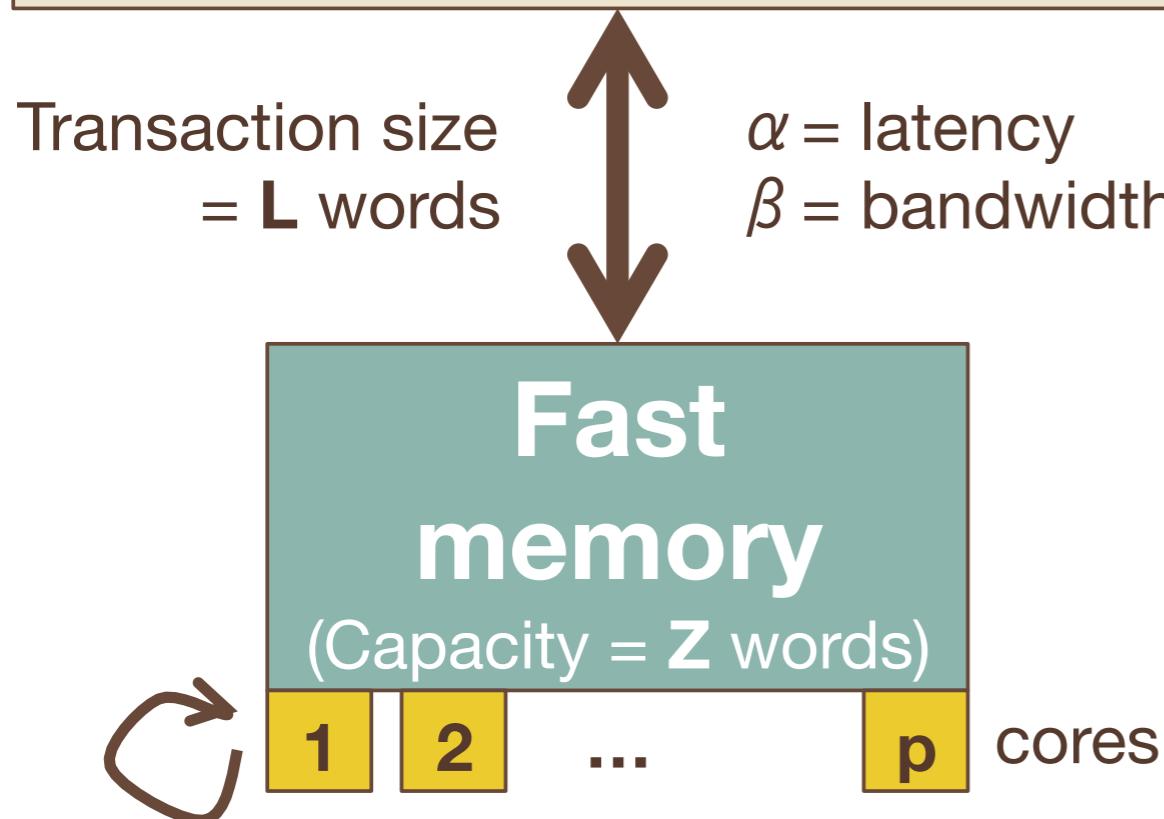
$$Q_p(n; Z, L) < Q_1(n; Z, L) + \mathcal{O}\left(\frac{p \cdot Z \cdot D(n)}{L}\right)$$

Example: Parallel depth-first + all-cores shared caches.

$$Q_p(n; Z + p \cdot L \cdot D(n), L) < Q_1(n; Z, L)$$

These design measures are
abstract. Can we **concretely**
estimate **time** on some target
architectural design?

Slow memory



$$C_0 = \text{flops} / \text{time} / \text{core}$$

~ Fermi (0th order)

e.g., Lee, Lakshminarayana, Kim, Vuduc (MICRO 2010)

$$T_{\text{mem}}(n)$$

$$= \alpha D(n) + \frac{Q_p(n; Z, L) \cdot L}{\beta}$$

$$T_{\text{comp}}(n)$$

$$= \left[D(n) + \frac{W(n)}{p} \right] \cdot \frac{1}{C_0}$$

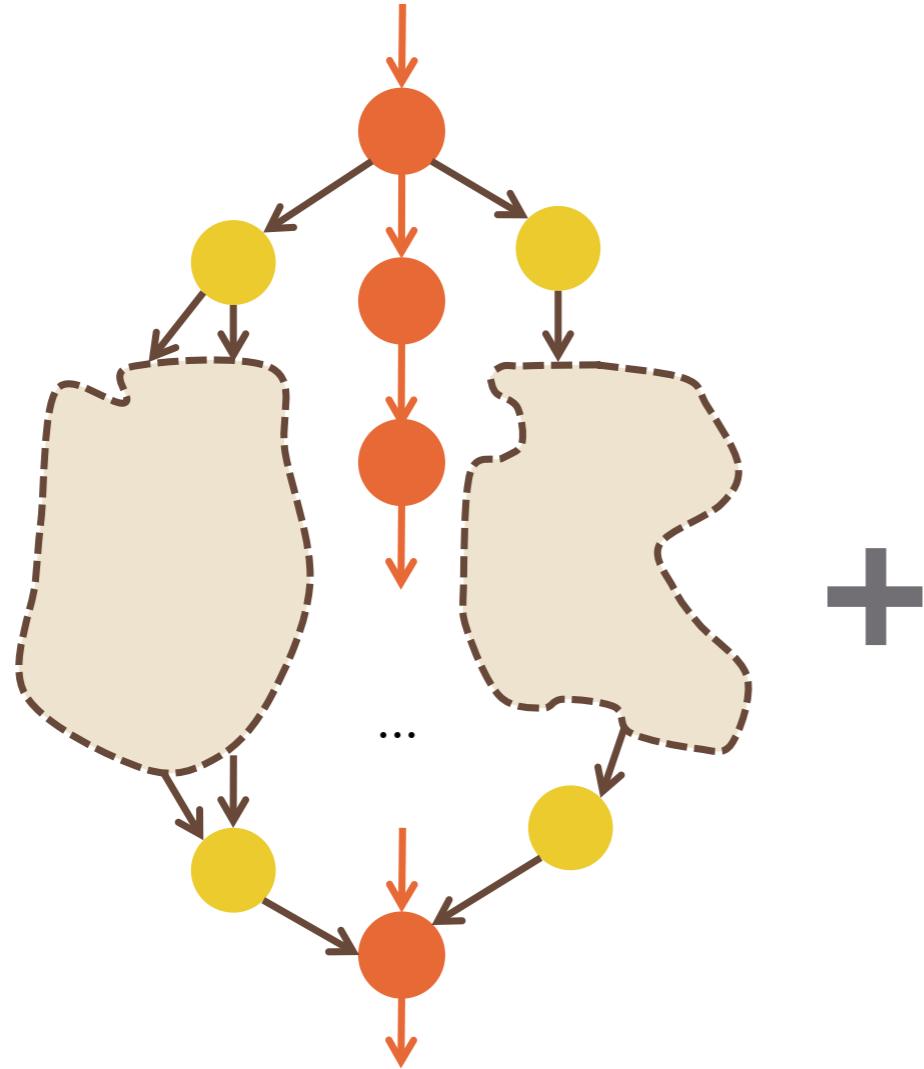
Brent (1975)

The preceding analysis gives us a way to connect **algorithms** and **architectures** explicitly.

Example: **Balance**.

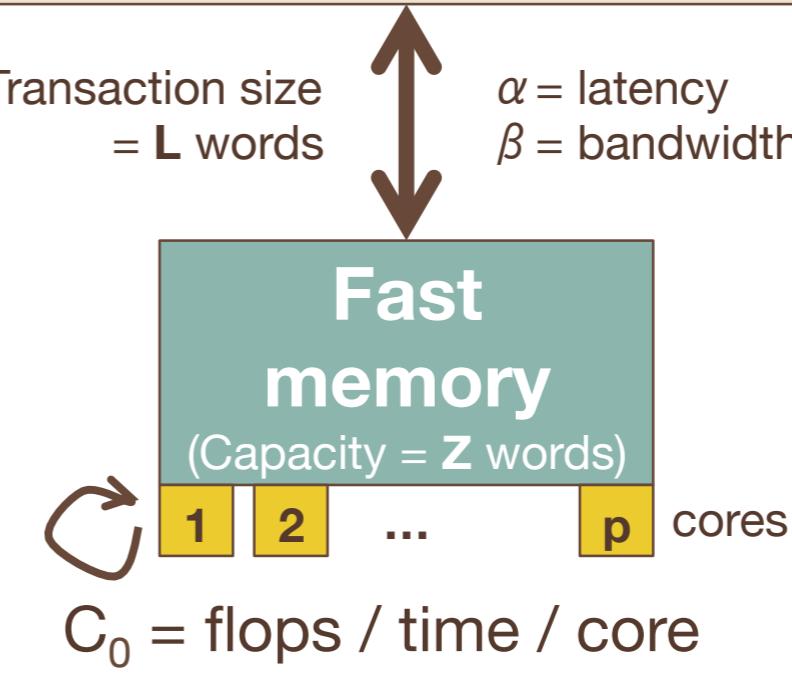
$$T_{\text{mem}}(n) \leq T_{\text{comp}}(n)$$

Kung (1986); Callahan, Cocke, Kennedy (1988); McCalpin (1995)

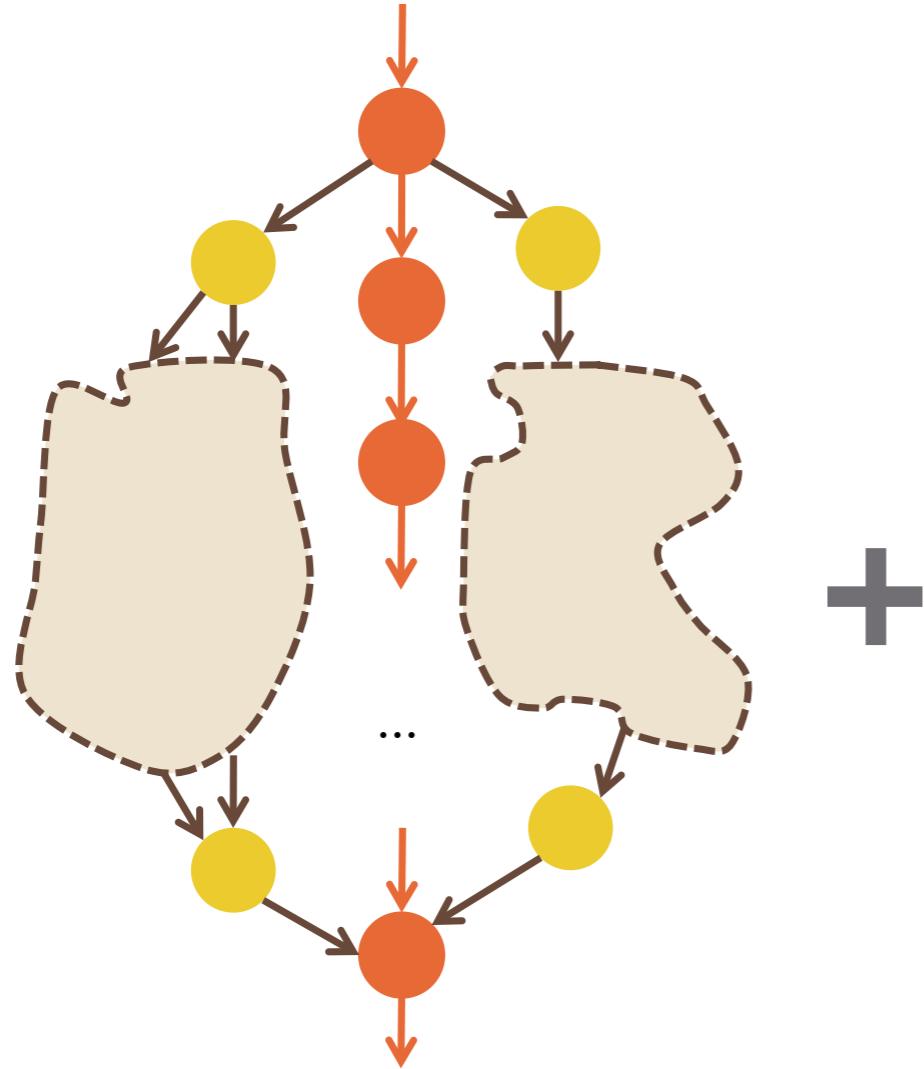


Slow memory

Transaction size
= L words α = latency
 β = bandwidth



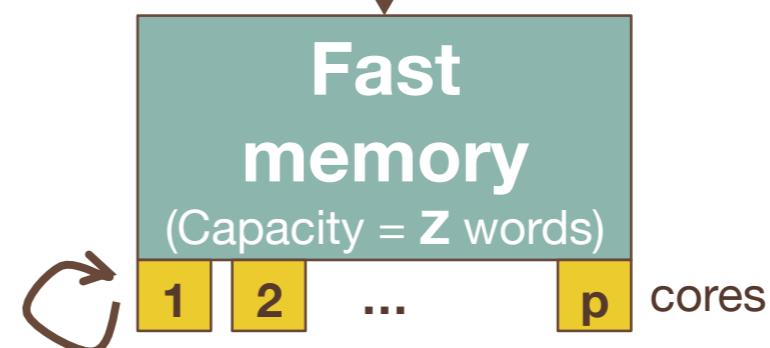
$$\frac{p \cdot C_0}{\beta/L} \left(1 + \frac{\alpha\beta/L}{Q_{p;Z,L}/D} \right) \leq \frac{W}{Q_{p;Z,L}L} \left(1 + \frac{p}{W/D} \right)$$



Slow memory

Transaction size
= L words

α = latency
 β = bandwidth

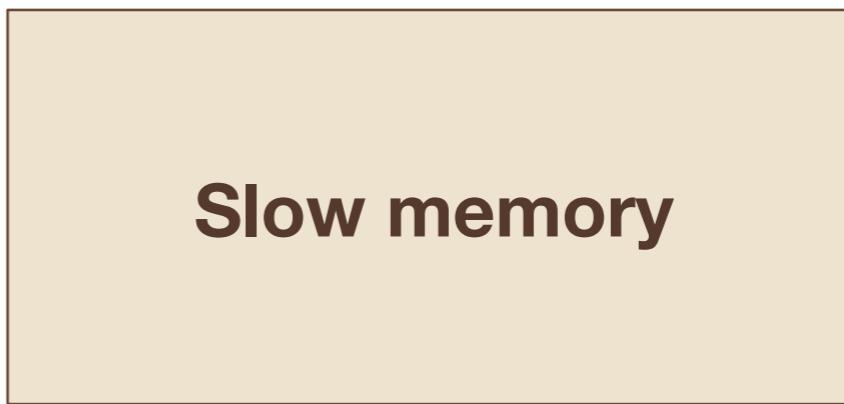
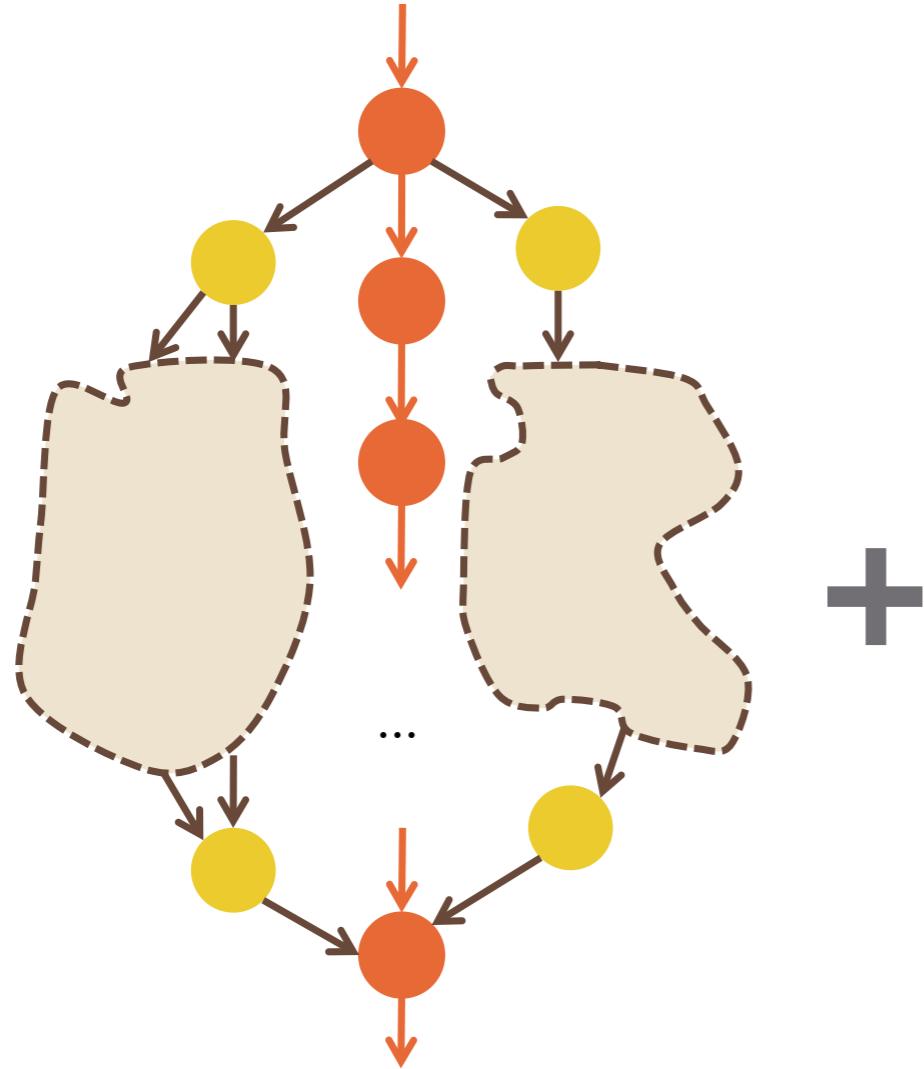


$C_0 = \text{flops} / \text{time} / \text{core}$

$$\frac{p \cdot C_0}{\beta/L} \left(1 + \frac{\alpha\beta/L}{Q_{p;Z,L}/D} \right) \leq \frac{W}{Q_{p;Z,L}L} \left(1 + \frac{p}{W/D} \right)$$

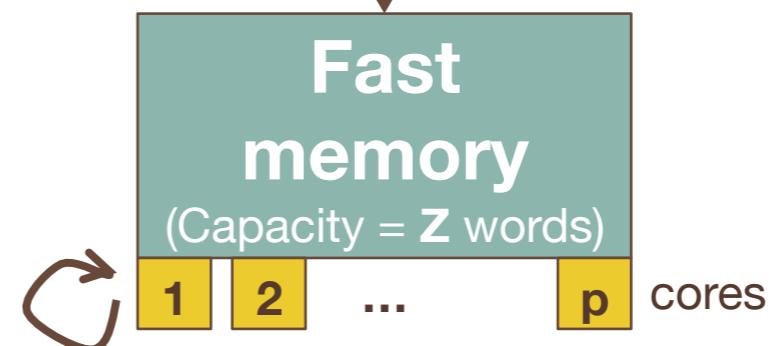


Balance



Transaction size
= L words

α = latency
 β = bandwidth



C_0 = flops / time / core

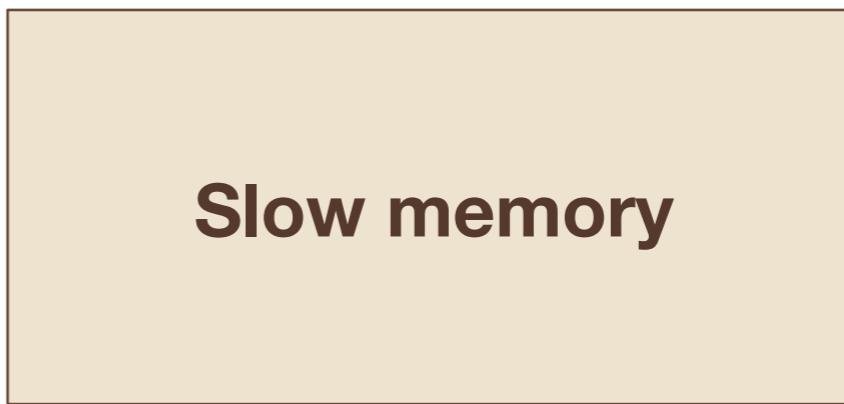
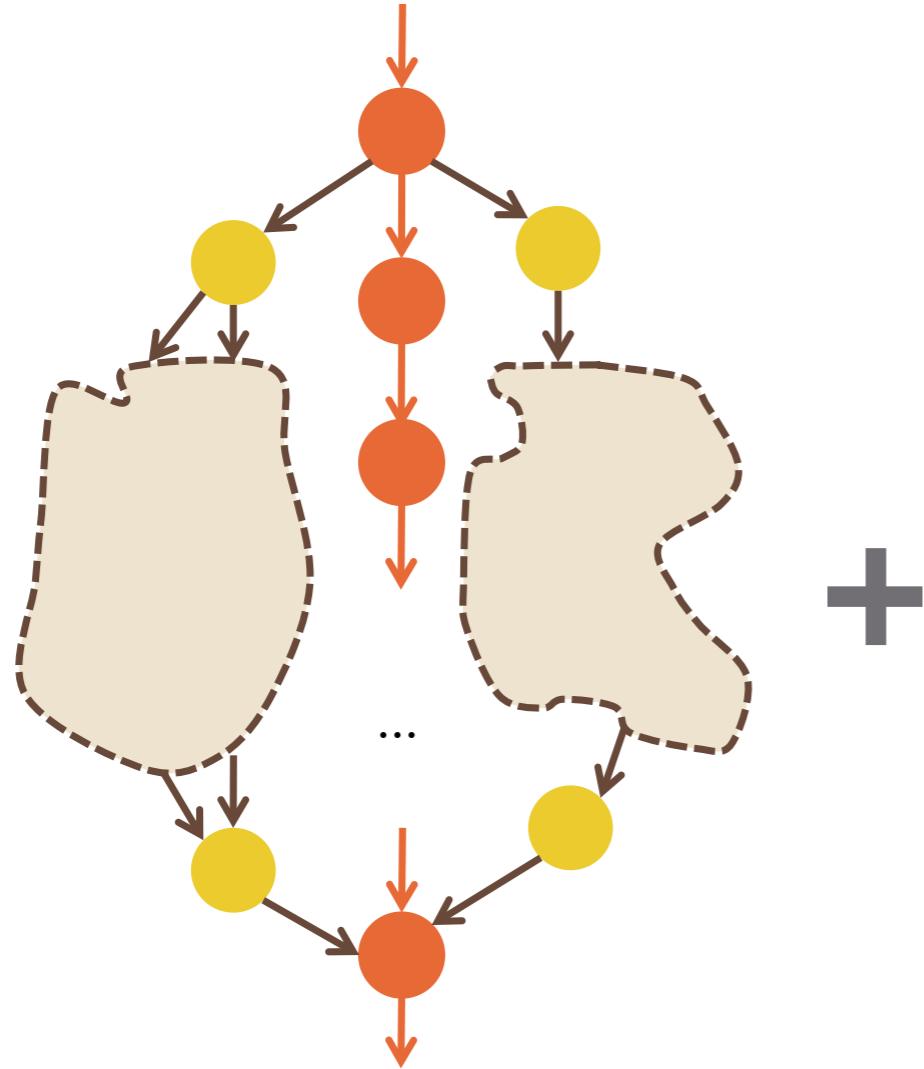
$$\frac{p \cdot C_0}{\beta/L} \left(1 + \frac{\alpha\beta/L}{Q_{p;Z,L}/D} \right) \leq \frac{W}{Q_{p;Z,L}L} \left(1 + \frac{p}{W/D} \right)$$



Balance

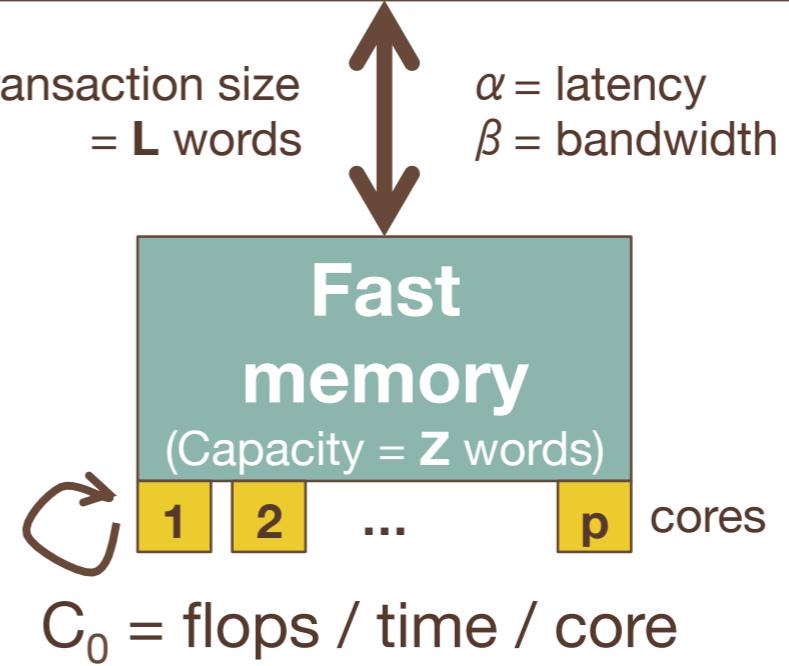


Intensity



Transaction size
= L words

α = latency
 β = bandwidth



$$C_0 = \text{flops} / \text{time} / \text{core}$$

$$\frac{p \cdot C_0}{\beta/L} \left(1 + \frac{\alpha\beta/L}{Q_{p;Z,L}/D} \right) \leq \frac{W}{Q_{p;Z,L}L} \left(1 + \frac{p}{W/D} \right)$$



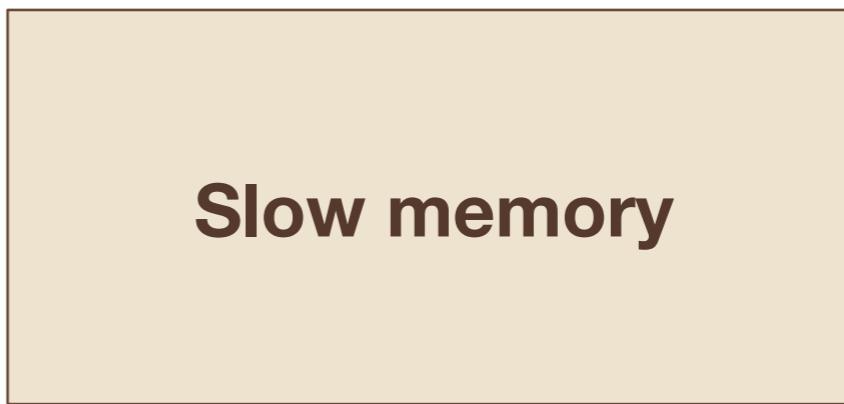
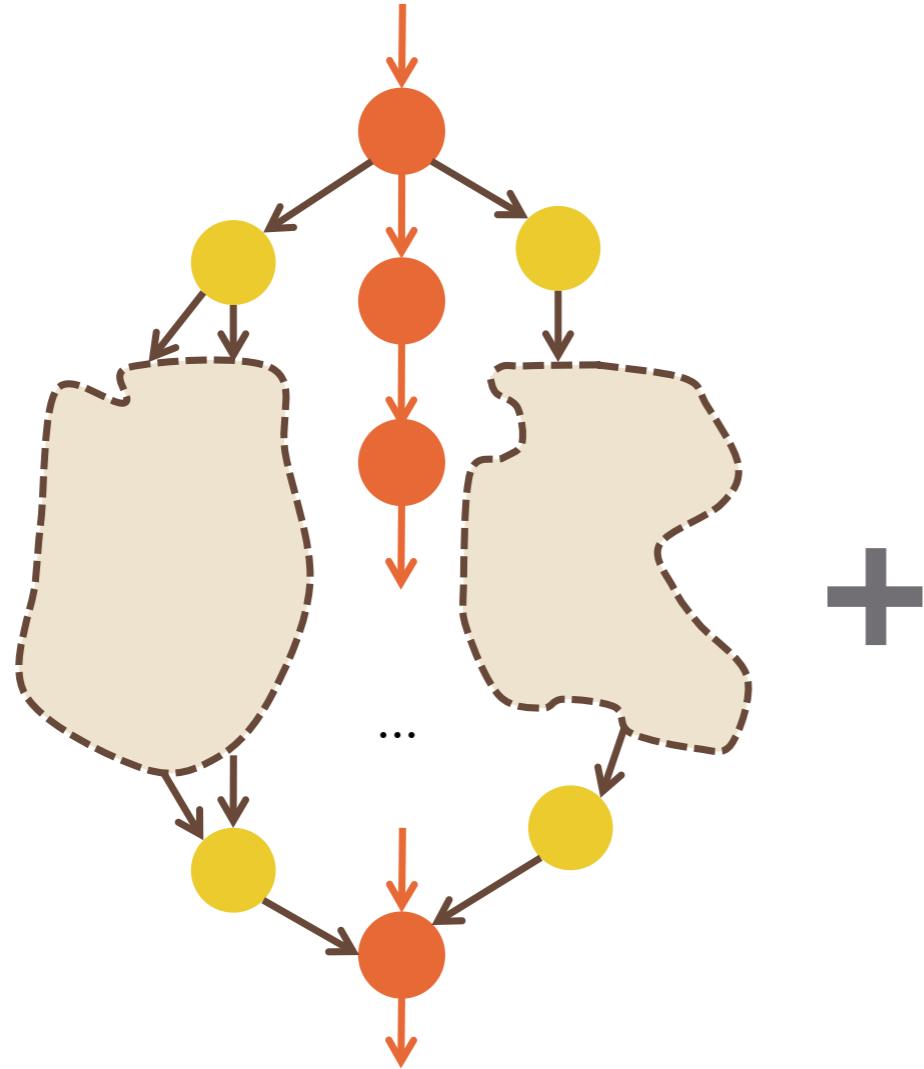
Balance



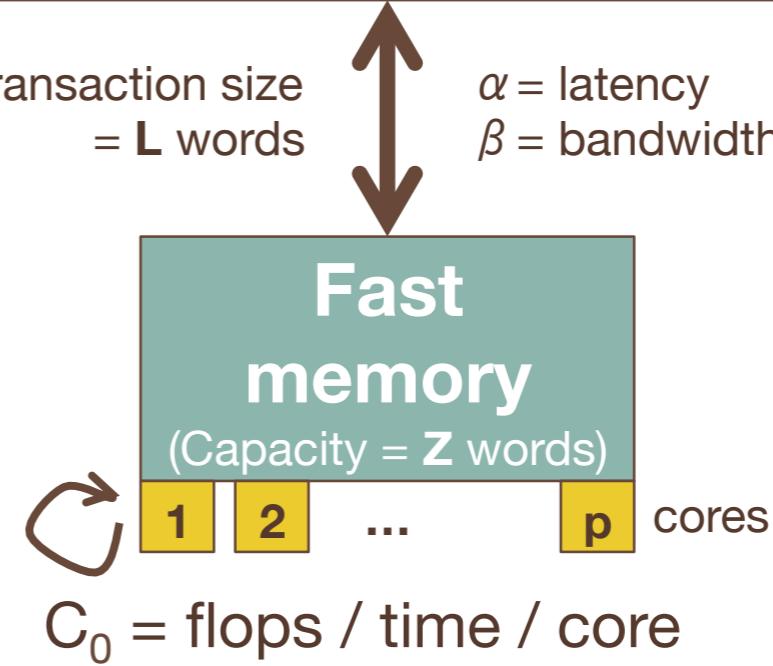
Little's Law



Intensity



Transaction size
= L words α = latency
 β = bandwidth



$$C_0 = \text{flops} / \text{time} / \text{core}$$

$$\frac{p \cdot C_0}{\beta/L} \left(1 + \frac{\alpha\beta/L}{Q_{p;Z,L}/D} \right) \leq \frac{W}{Q_{p;Z,L}L} \left(1 + \frac{p}{W/D} \right)$$



Balance



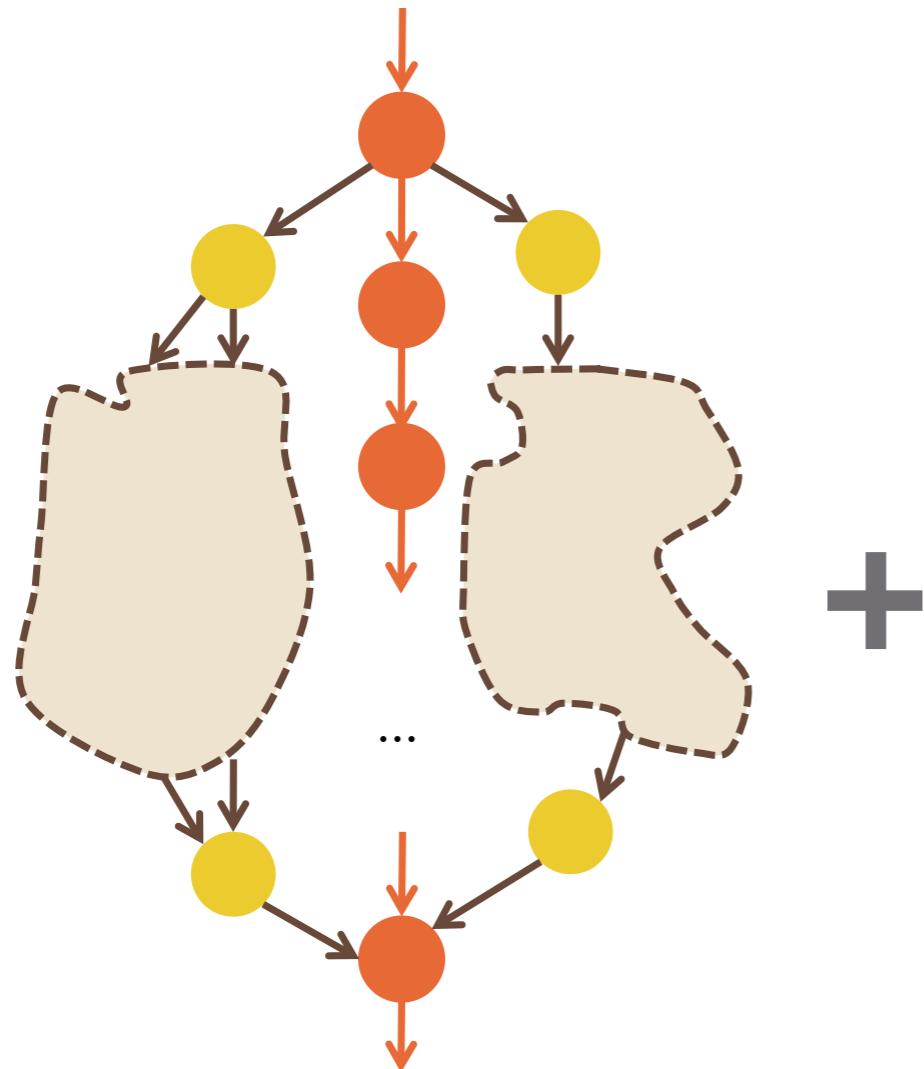
Little's Law



Intensity

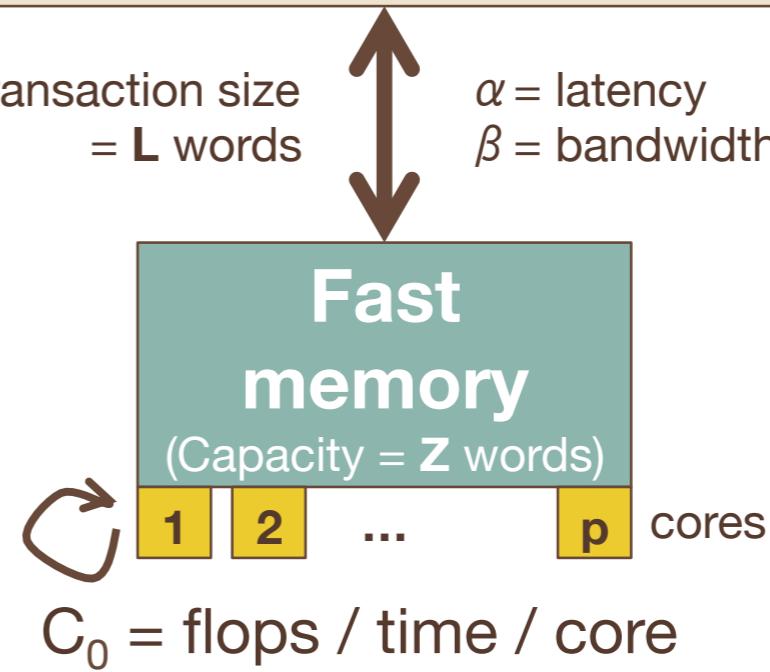


Amdahl's Law



Slow memory

Transaction size
= L words $\alpha = \text{latency}$
 $\beta = \text{bandwidth}$



$$\frac{p \cdot C_0}{\beta} \leq \mathcal{O}\left(\sqrt{\frac{Z}{p}}\right)$$

Example: **Matrix-multiply + work-stealing**

flop : byte

40

30

20

10

Intensity

Balance

2010

2012

2014

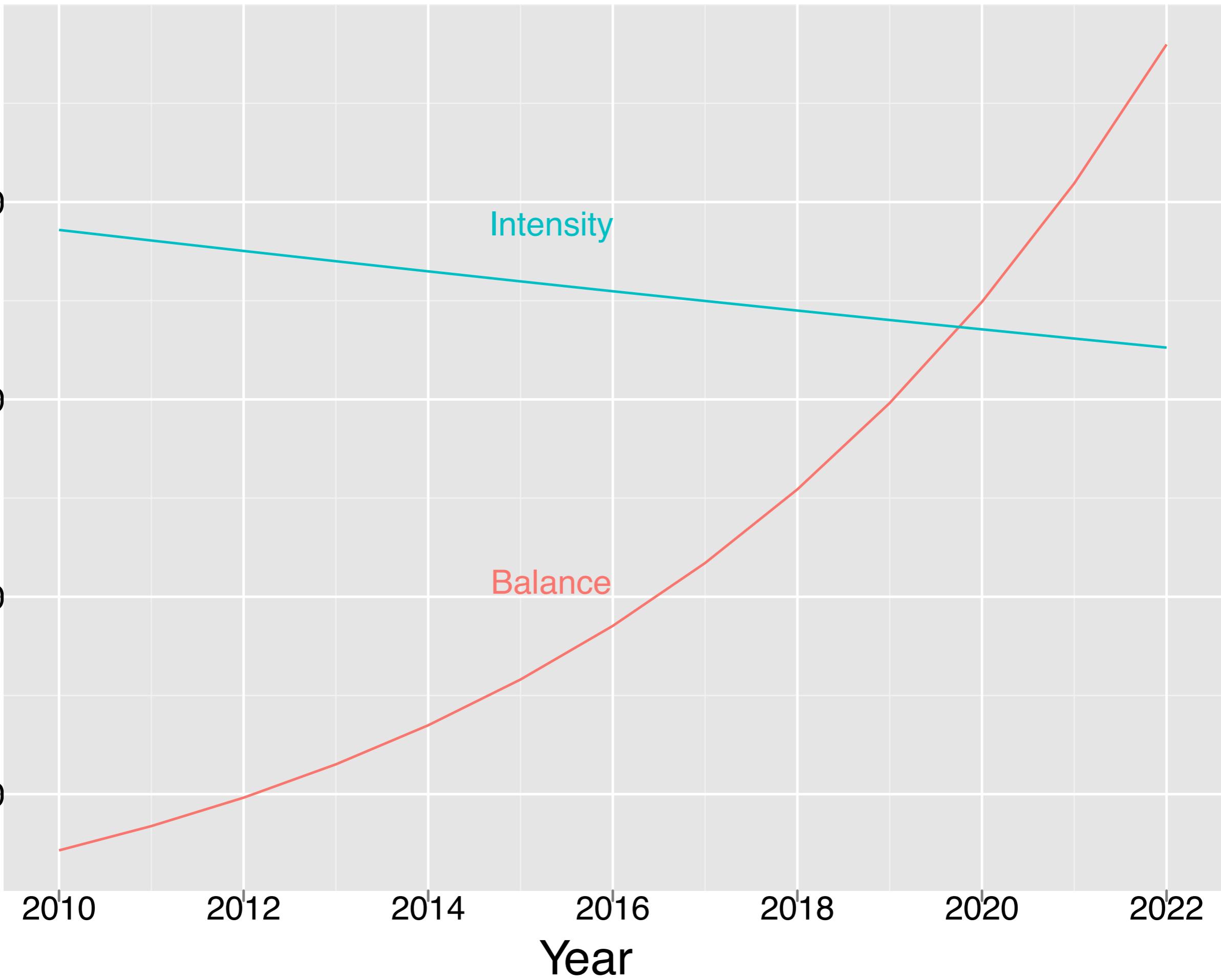
2016

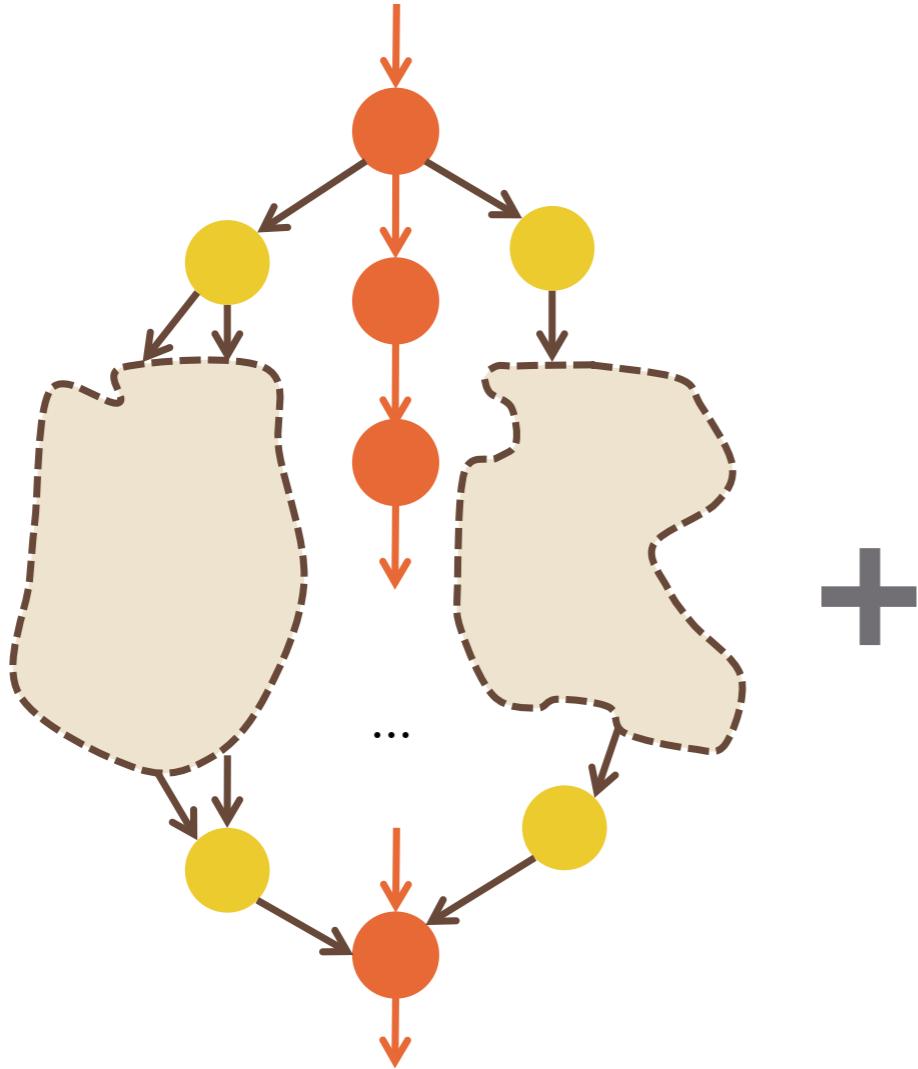
2018

2020

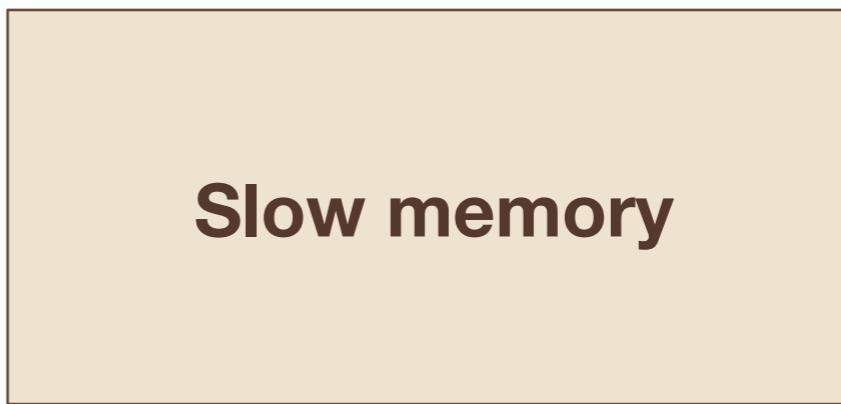
2022

Year

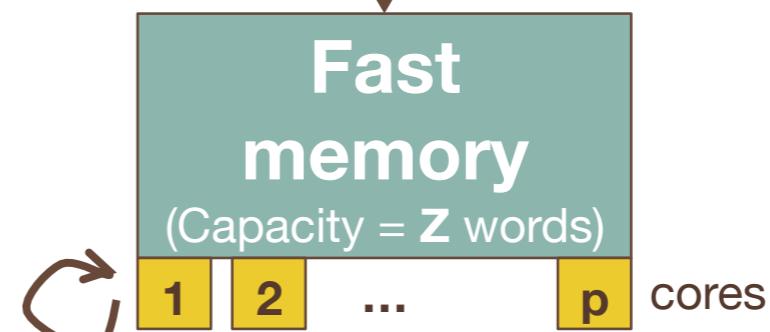




+



Transaction size
= L words $\alpha = \text{latency}$
 $\beta = \text{bandwidth}$



$$C_0 = \text{flops} / \text{time} / \text{core}$$

$$\frac{p \cdot C_0}{\beta} \leq \mathcal{O} \left(\log \frac{Z}{p} \right)$$

Example: Cache-oblivious comparison-based sorting* + work-stealing

Summary – How principles inform practice:

1. Make algorithm-architecture co-design precise:
**Quantitative analysis + predictions,
with one big caveat: constants!**

**Balance work (not flops) & “life” (communication).
Algorithmic fine-grained asynchrony by default.
“Energy-aware,” for algorithms, is a red herring.**
2. Suggest performance analysis metrics.
3. Imply a programming style.

Question: How do principles inform practice?

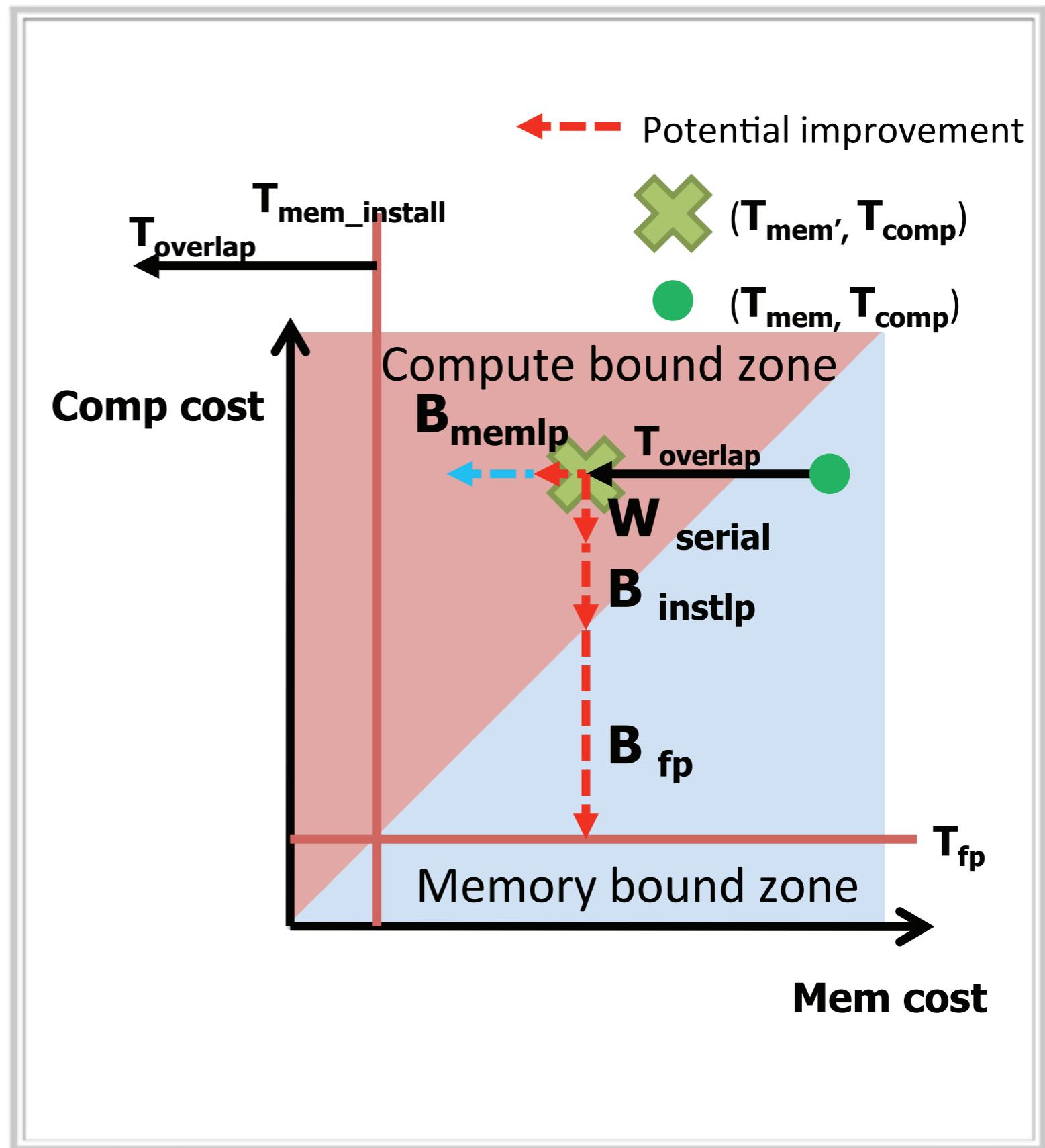
Answers (?):

1. Make algorithm-architecture co-design precise
2. **Suggest performance analysis metrics**
3. Imply a programming style

On-going work:

Marrying the high-level algorithmic model with low-level architectural models to build performance analysis tools for GPU platforms.

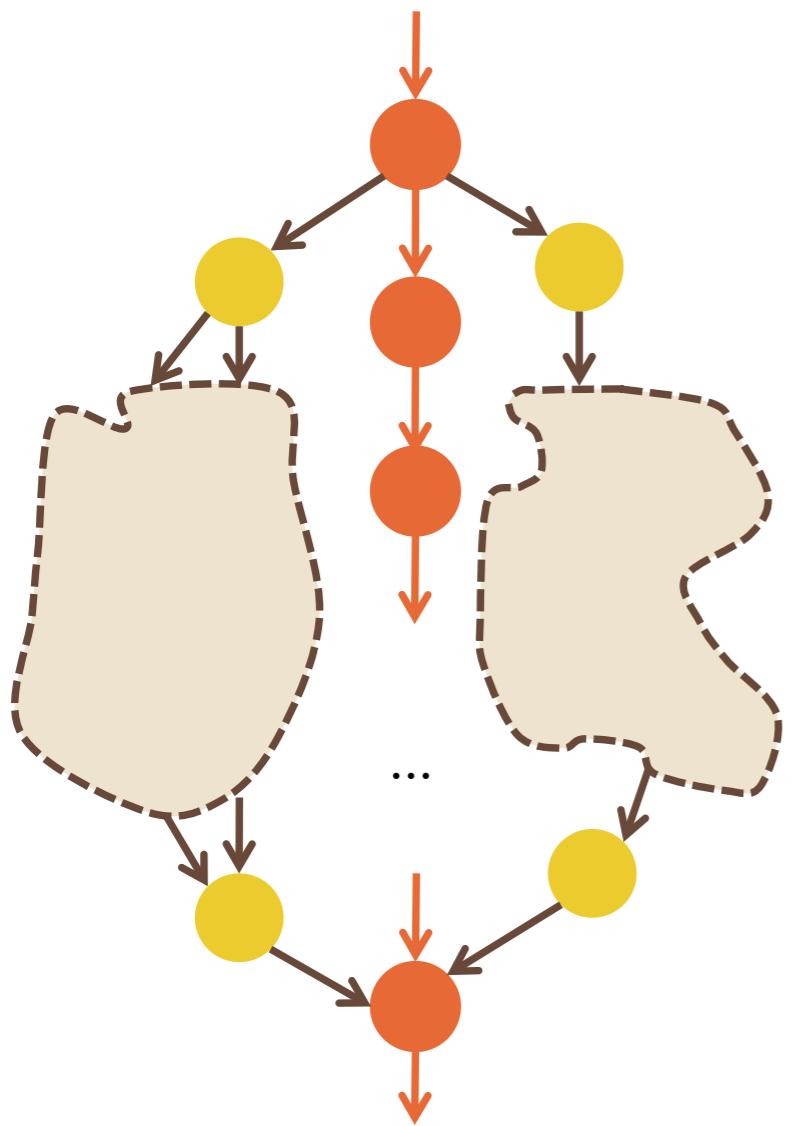
Joint with H. Kim @ GT



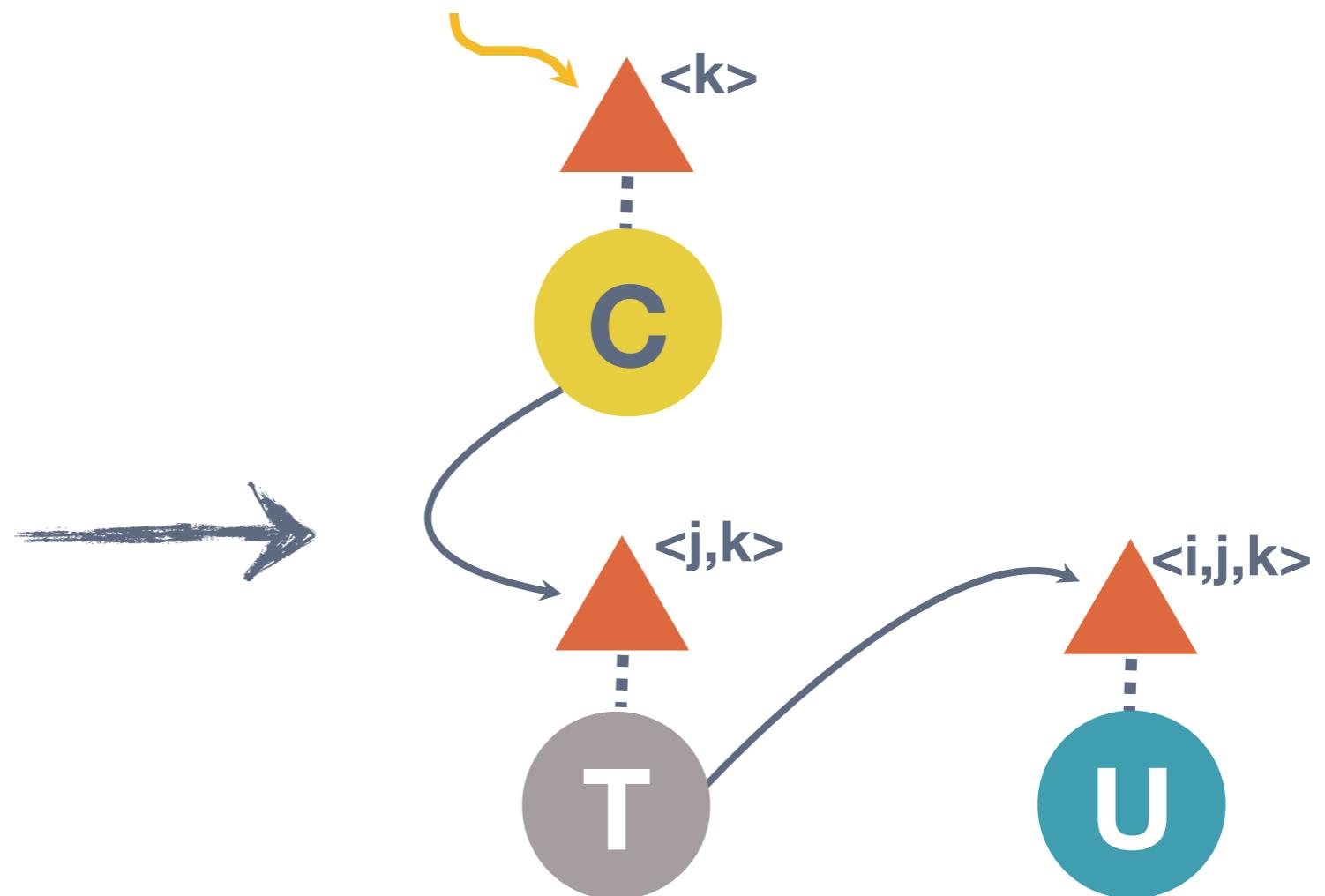
Question: How do principles inform practice?

Answers (?):

1. Make algorithm-architecture co-design precise
2. Suggest performance analysis metrics
3. **Imply a programming style**



CONCURRENT COLLECTIONS



Kath Knobe (Intel), Vivek Sarkar (Rice)

Our take? See Chandramowlishwaran et al.
IPDPS'10 “best paper” winner

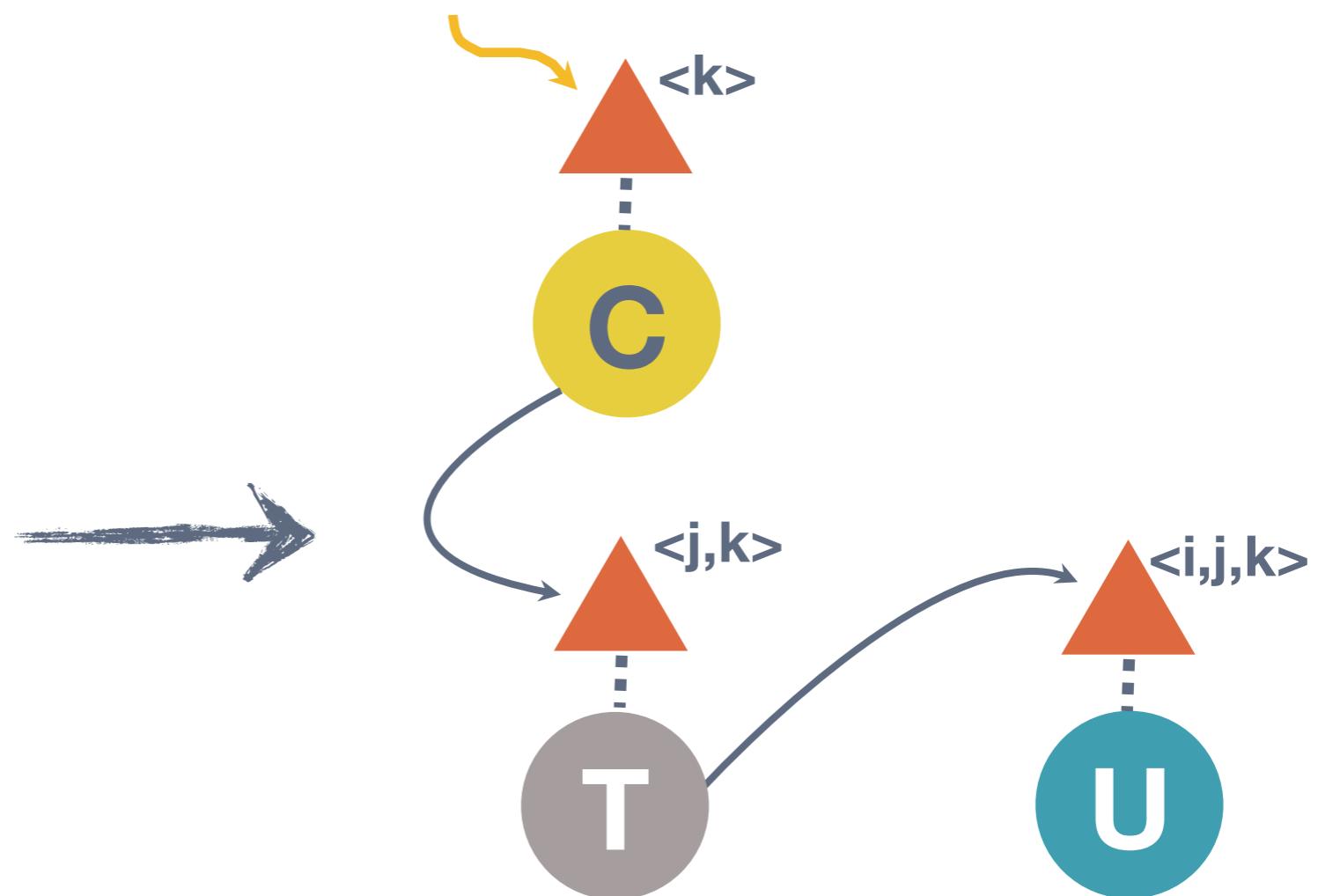
CnC is a *coordination* language (model).

Can use CnC as “glue” at multiple levels of parallelism — “beyond the software pipeline.”

Can retrofit CnC onto an existing code.

CnC’s program representation facilitates checkpointing.

CONCURRENT COLLECTIONS



Kath Knobe (Intel), Vivek Sarkar (Rice)

Our take? See Chandramowlishwaran et al.
IPDPS’10 “best paper” winner

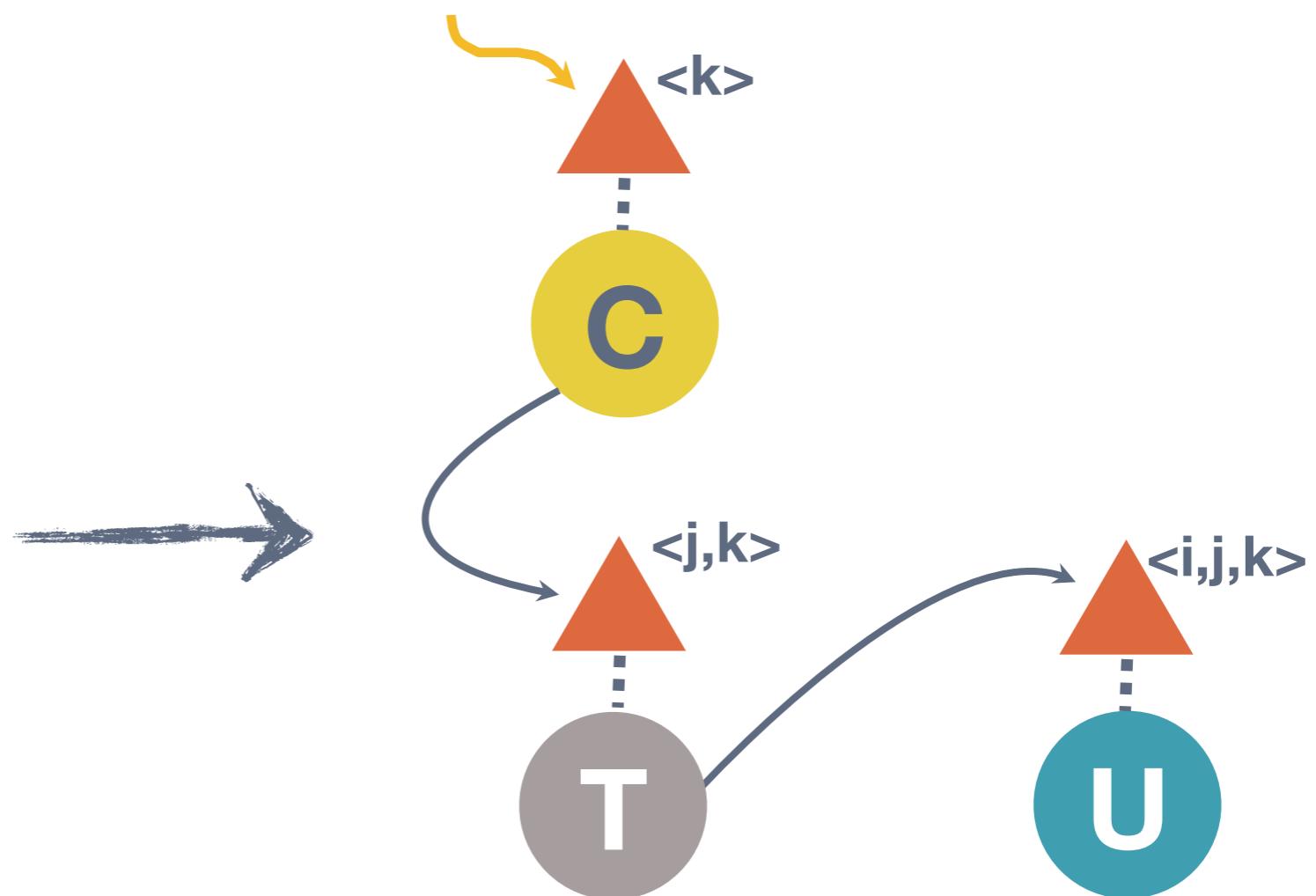
CnC is a *coordination* language (model).

Can use CnC as “glue” at multiple levels of parallelism — “beyond the software pipeline.”

Can retrofit CnC onto an existing code.

CnC’s program representation facilitates checkpointing.

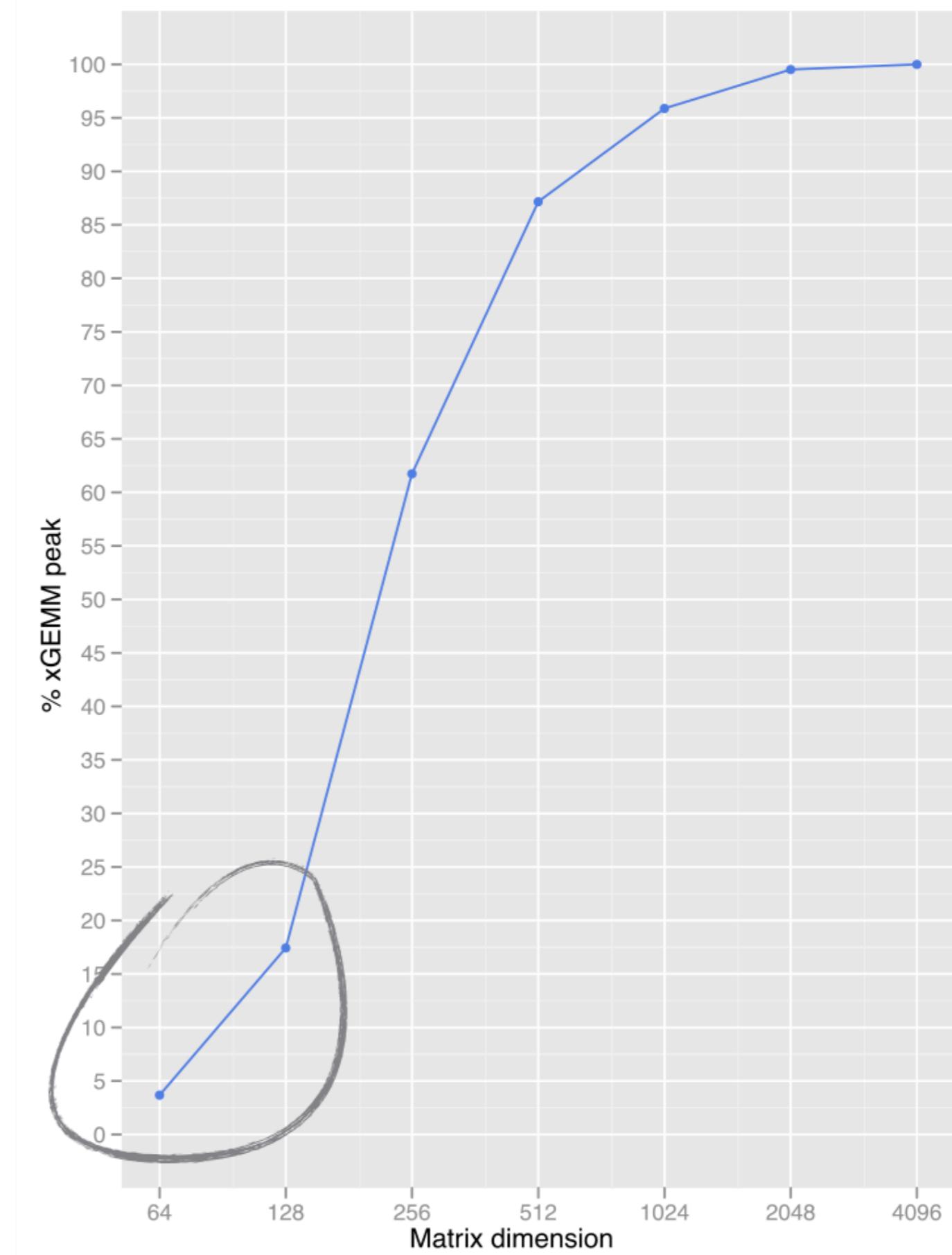
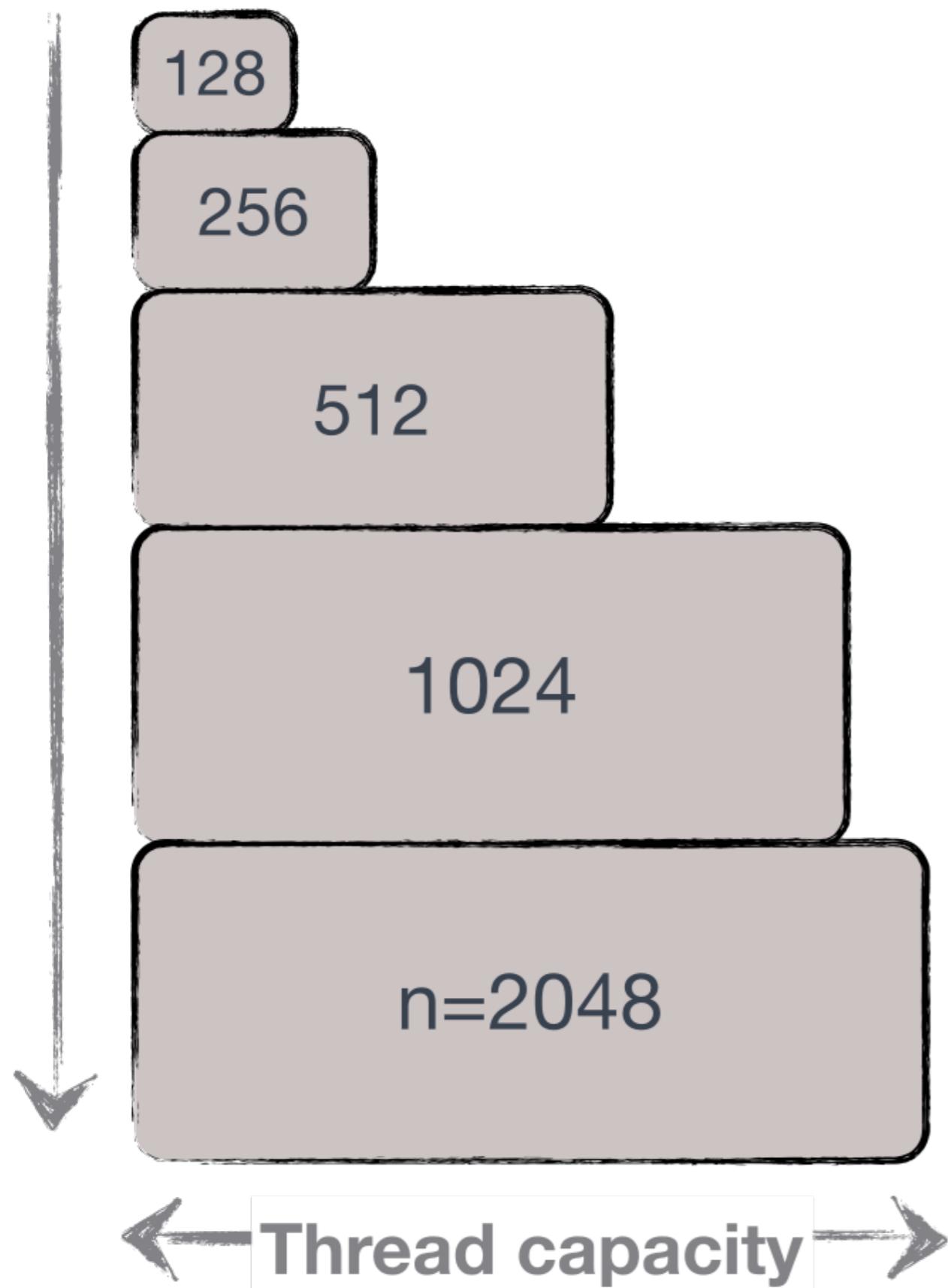
CONCURRENT
COLLECTIONS



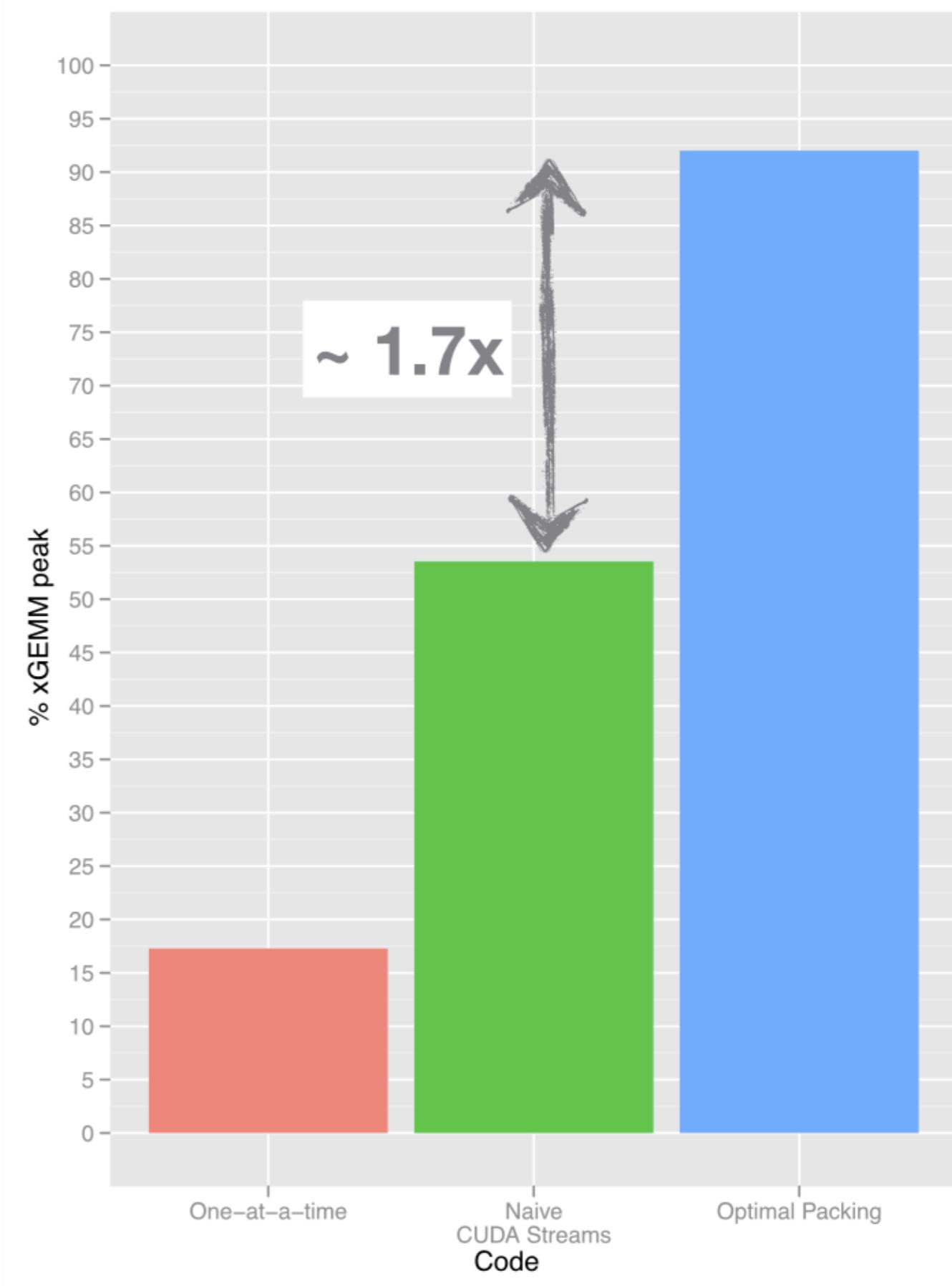
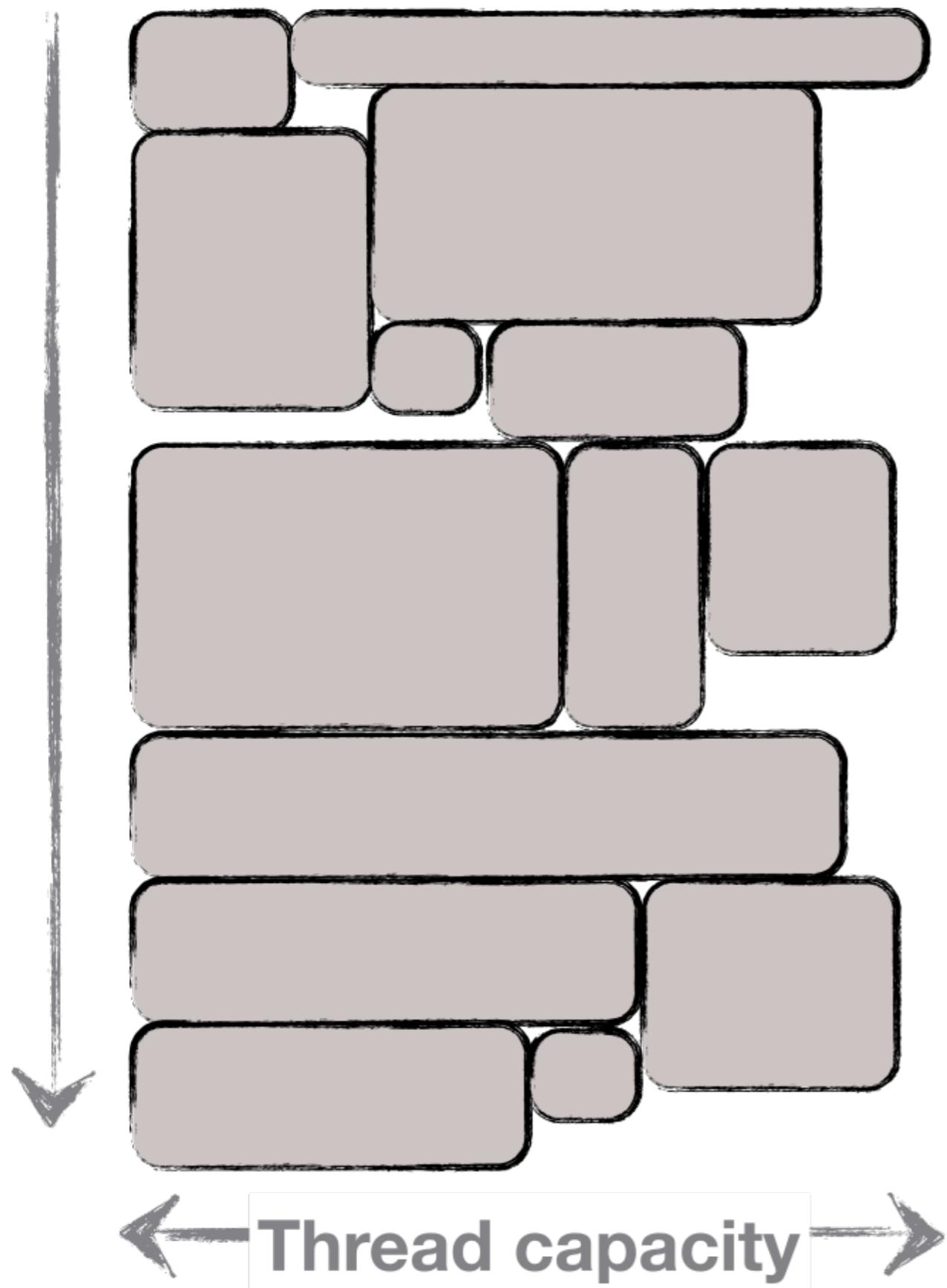
Kath Knobe (Intel), Vivek Sarkar (Rice)

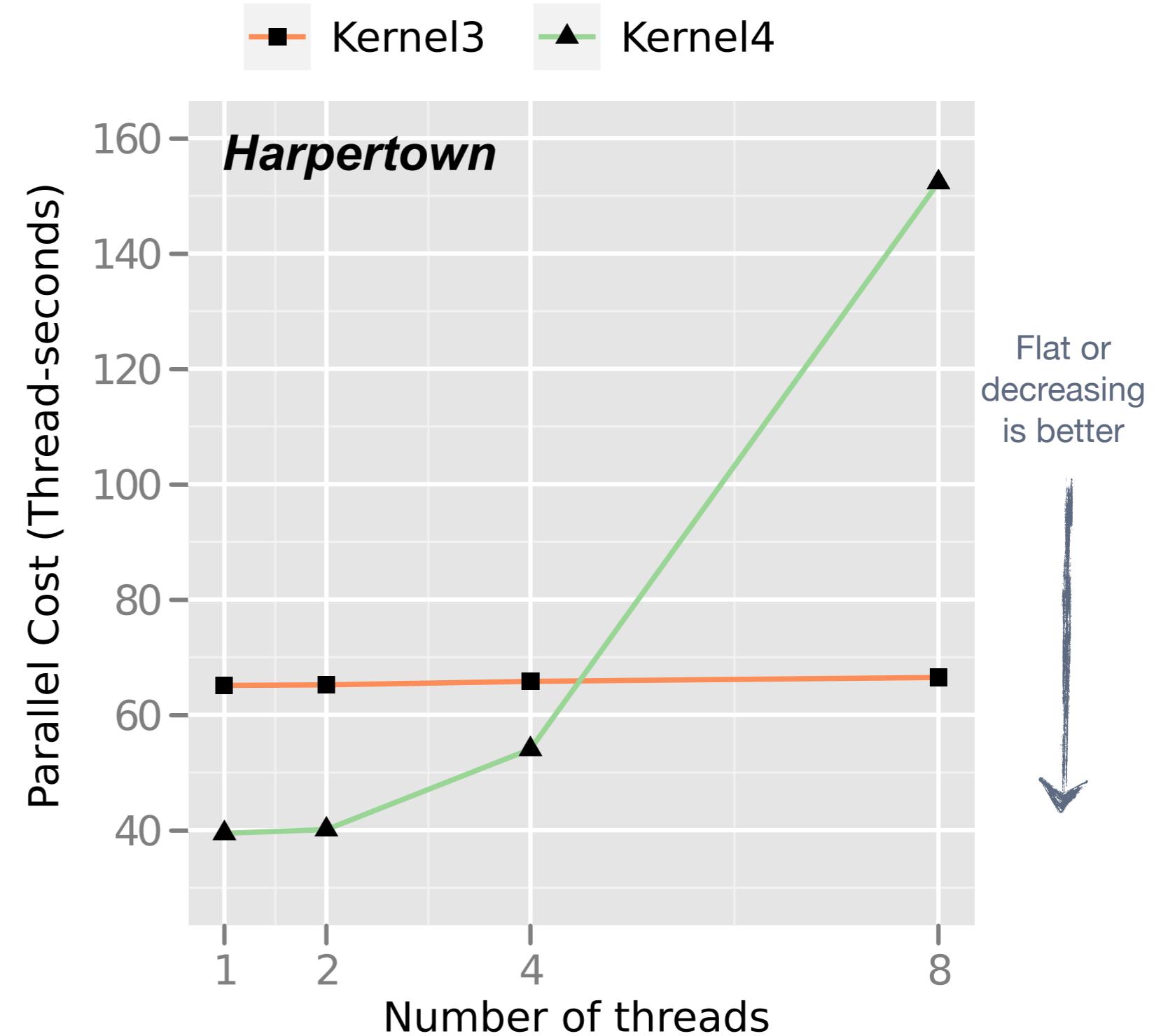
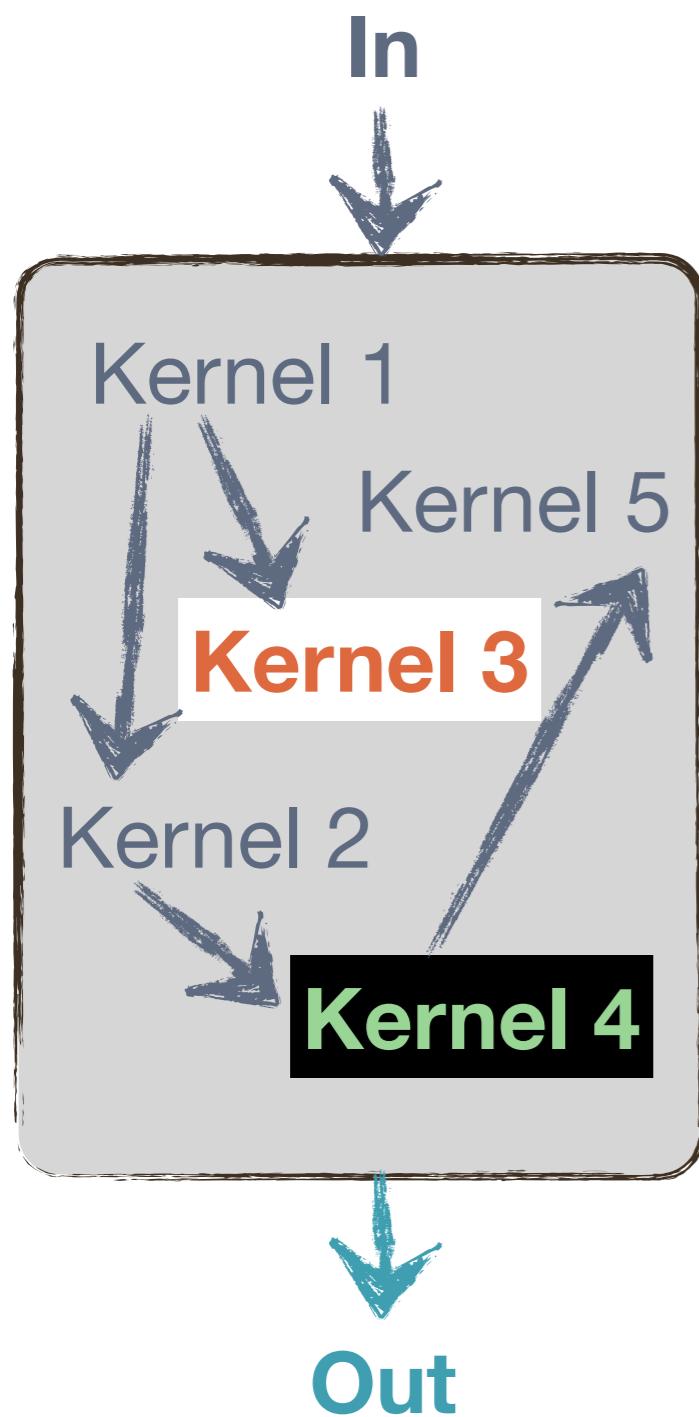
Extensions for temporally streaming apps by,
e.g., **K. Ramachandran (GT)**

Time



Time



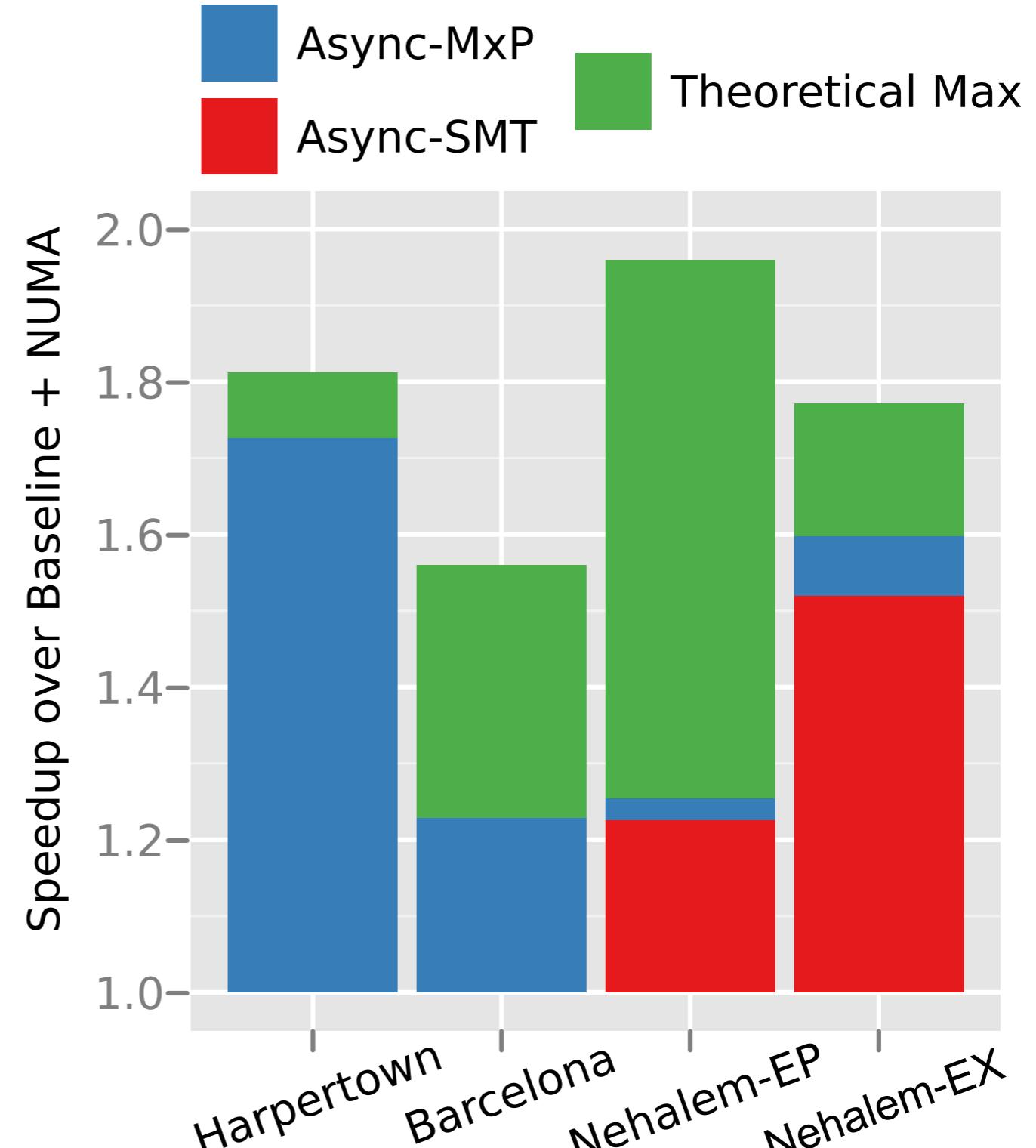


ASYNCHRONOUS BY DEFAULT

Assume additional knowledge of the task dependency structure

- Idea: Mixed phase execution
- That is, run K4 up to its scalability limit, and use remaining cores for K3

ASYNCHRONOUS BY DEFAULT

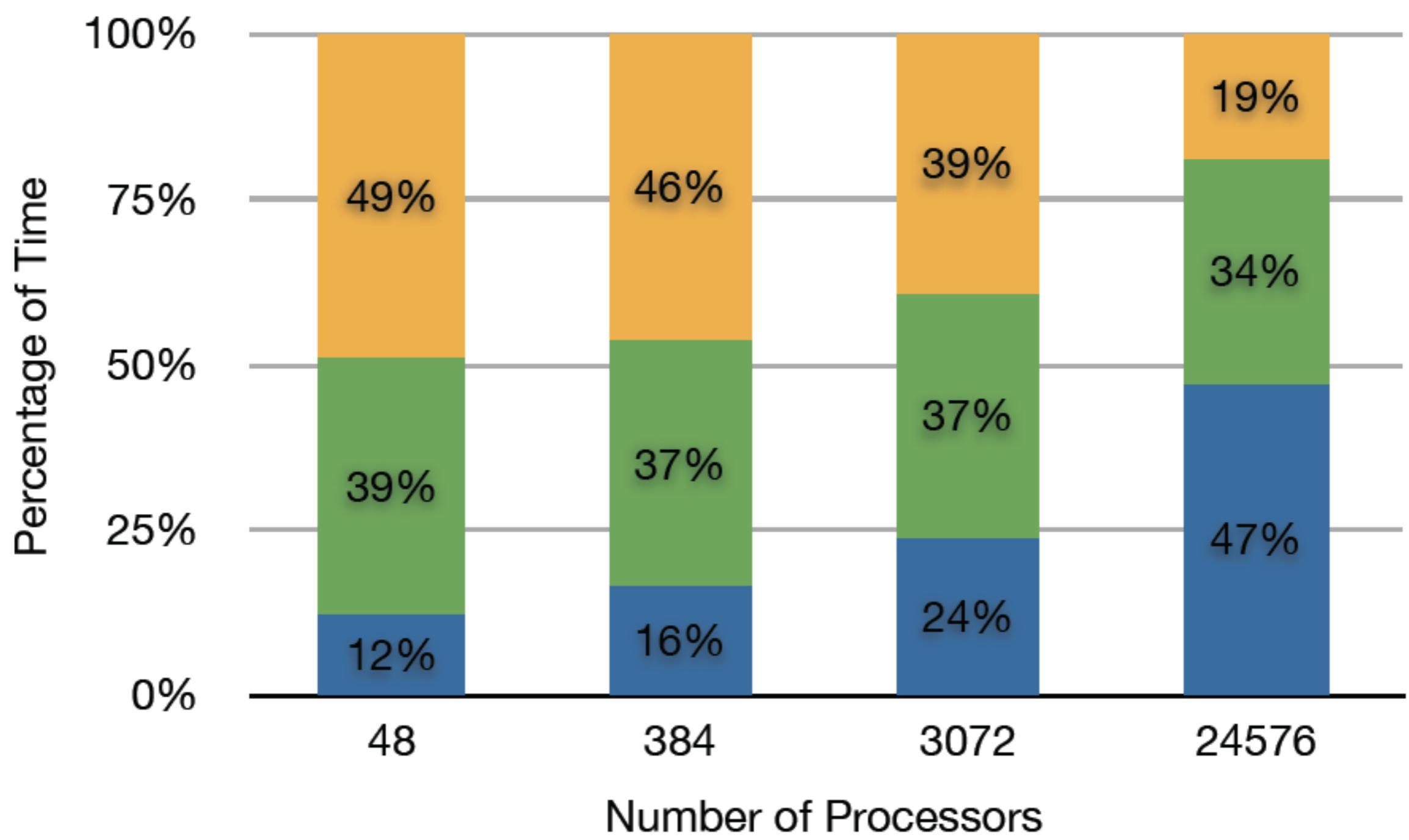


1.2–1.7x improvements possible, regardless of SMT availability

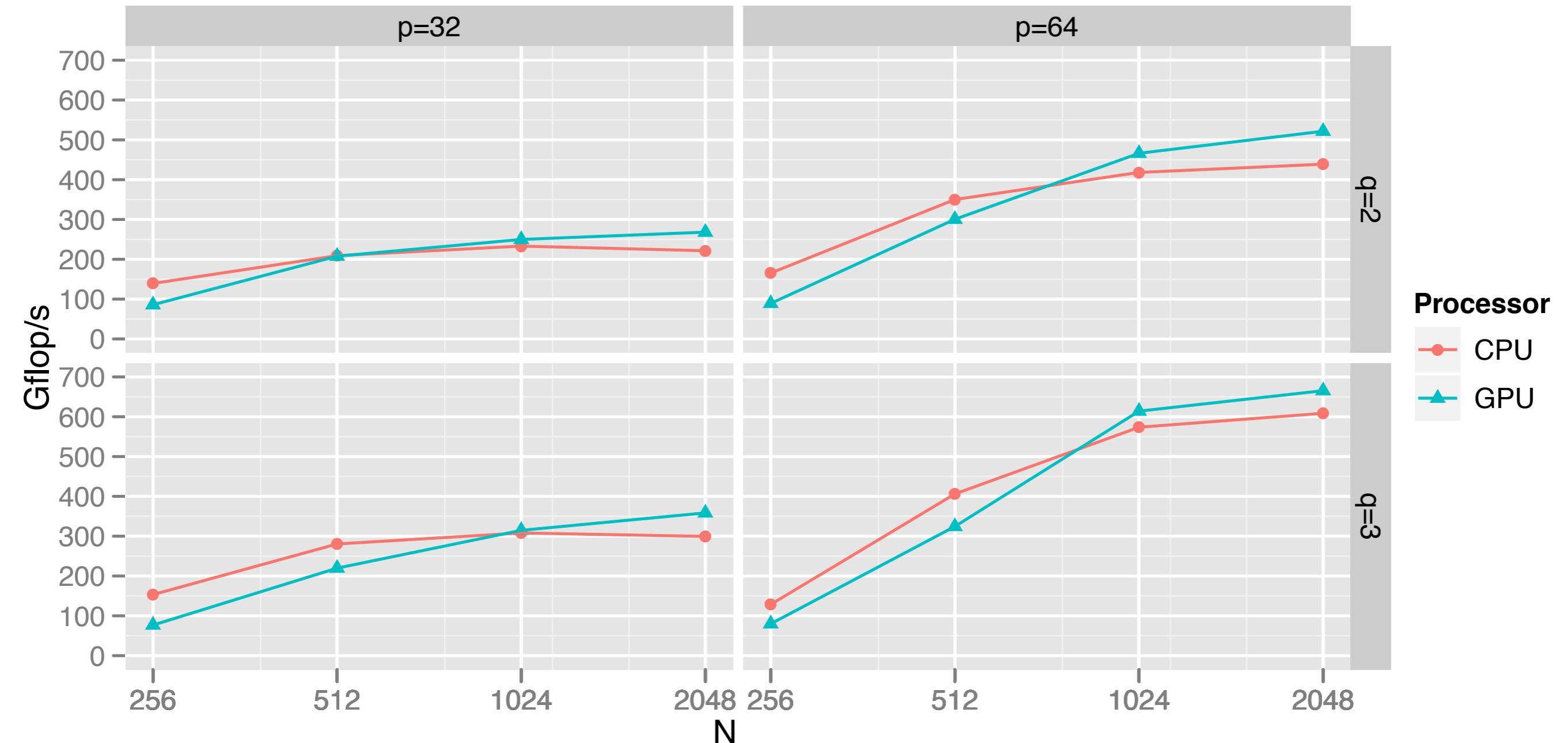
Summary – How principles inform practice:

1. Make algorithm-architecture co-design precise:
Quantitative analysis + predictions
2. Suggest performance analysis metrics:
**Work, depth, I/O, balance, intensity,
compute- vs. memory-level parallelism**
3. Imply a programming style:
DAGs + asynchrony by default

- Backup slides

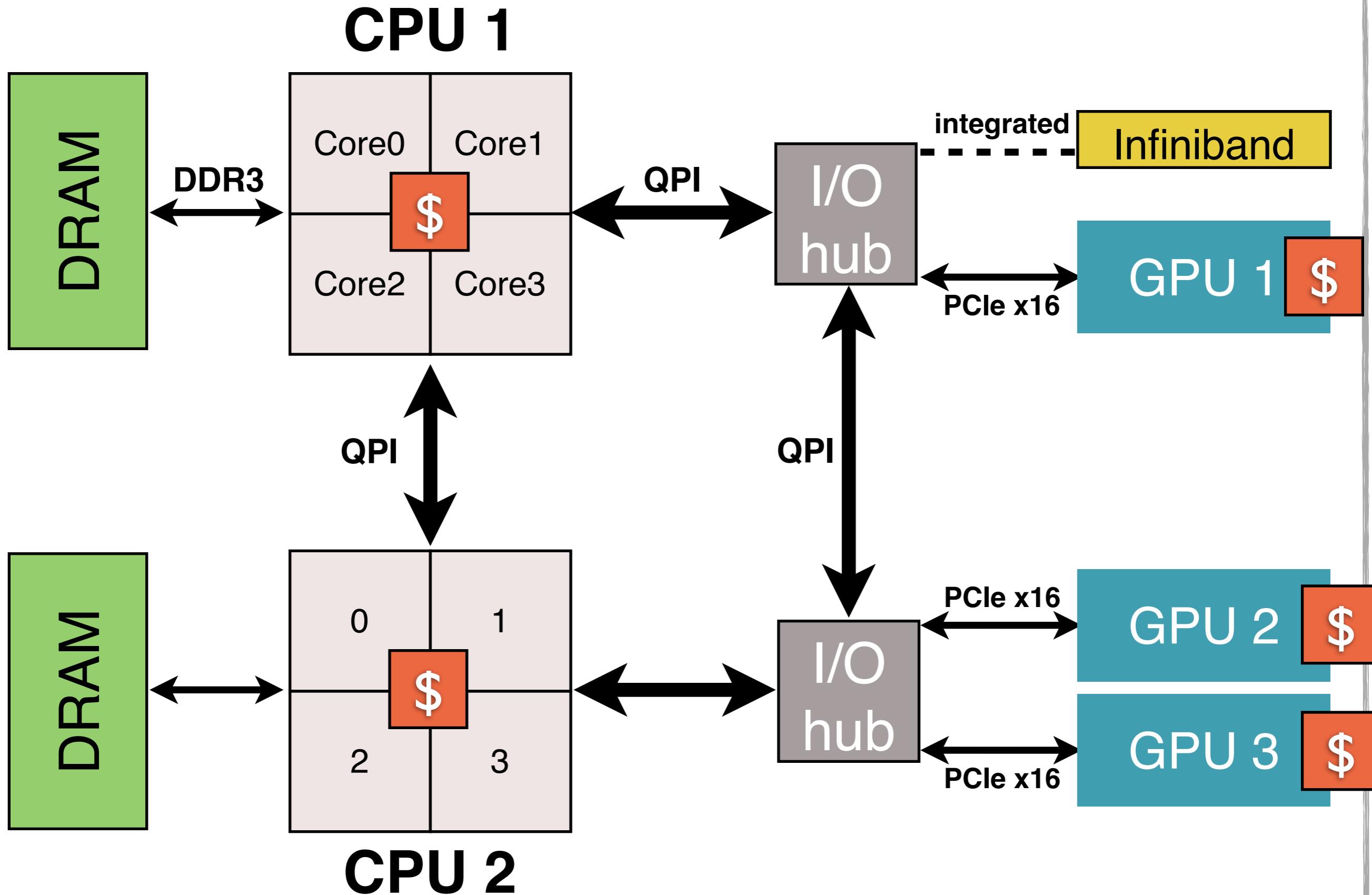


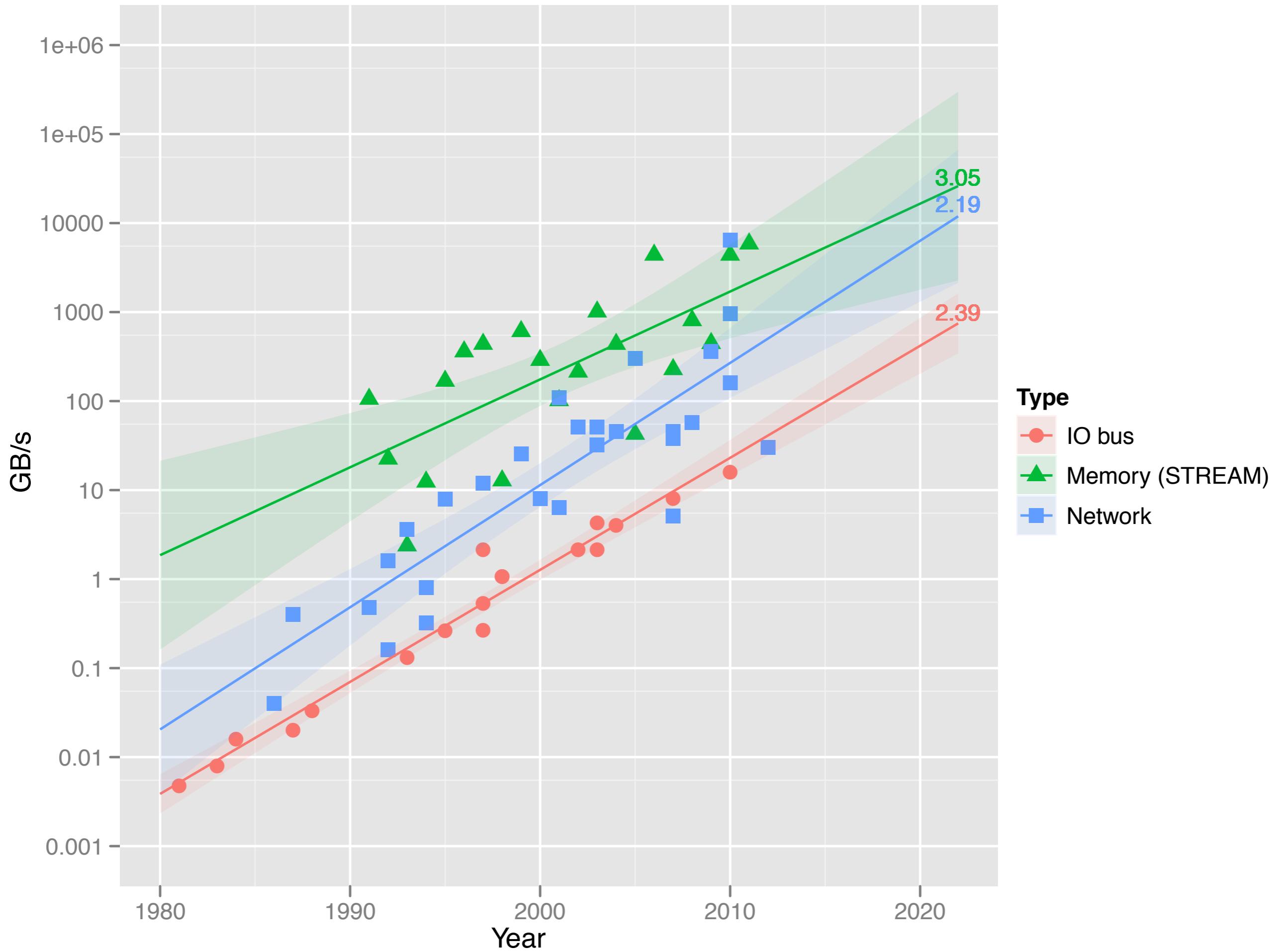
- Work, not flops (though includes flops)
- Emphasizes finding and minimizing the critical path
- Pedagogically unifies several performance engineering primitives, makes “compute” and “memory” parallelism explicit
- Directly relates machine parameters in algorithm-specific context (co-design)
- Asynchrony by default (DAG)
- Two exercises
 - Use the model to predict the future
 - Suggest what features a programming model should have

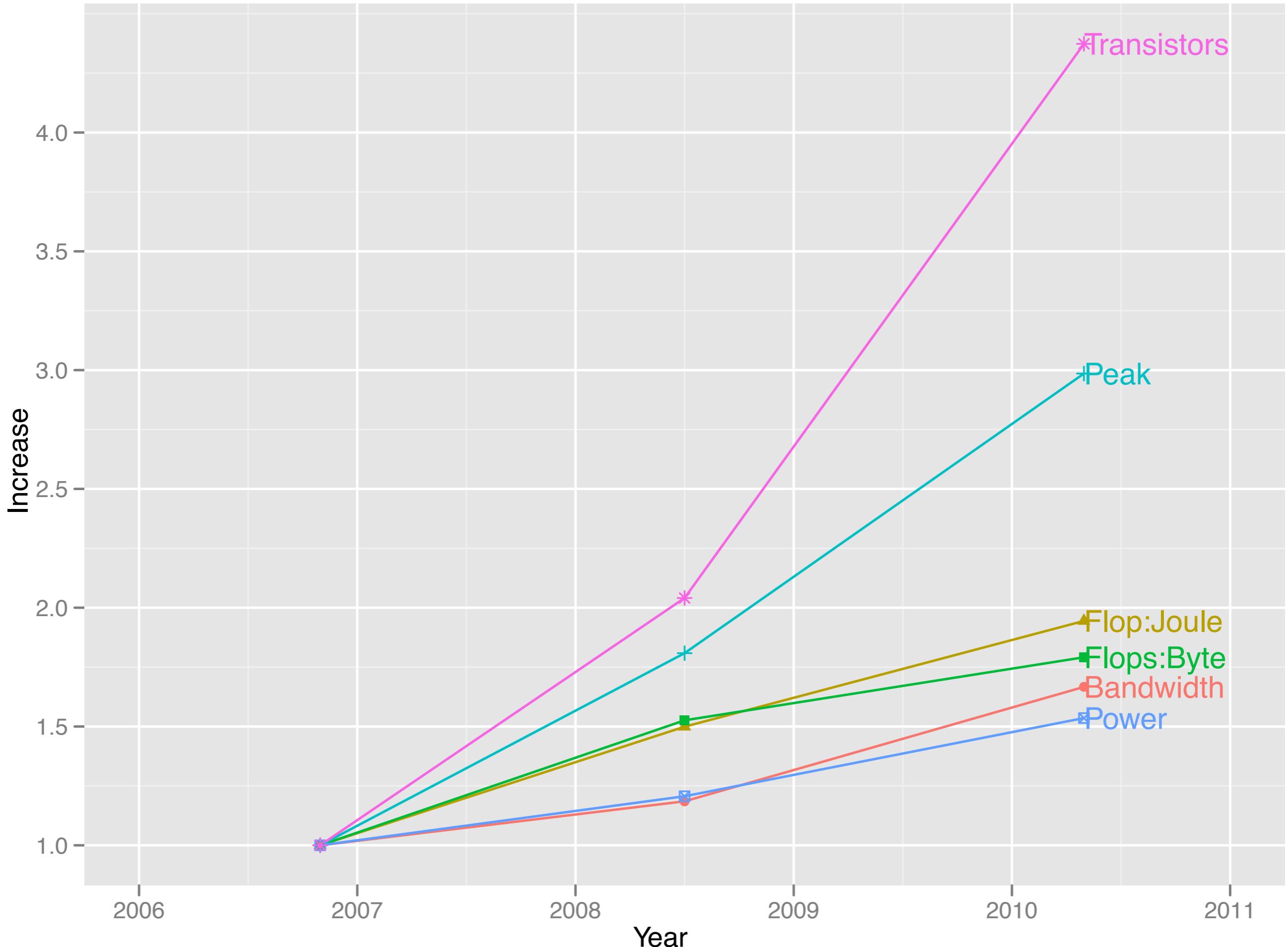


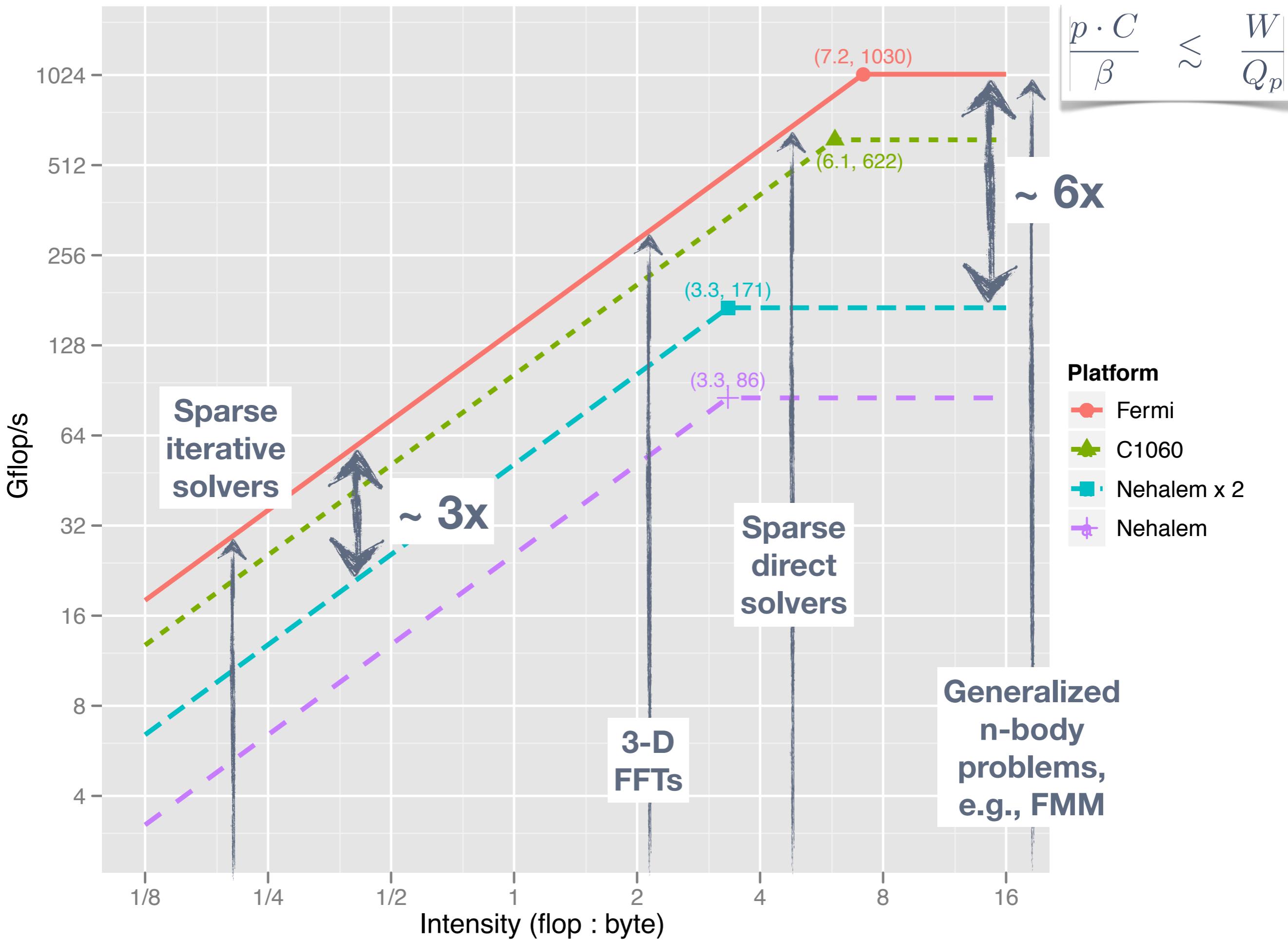
3D FFT, $N * N * N$ – P3DFFT, CUFFT vs. FFTW / MKL

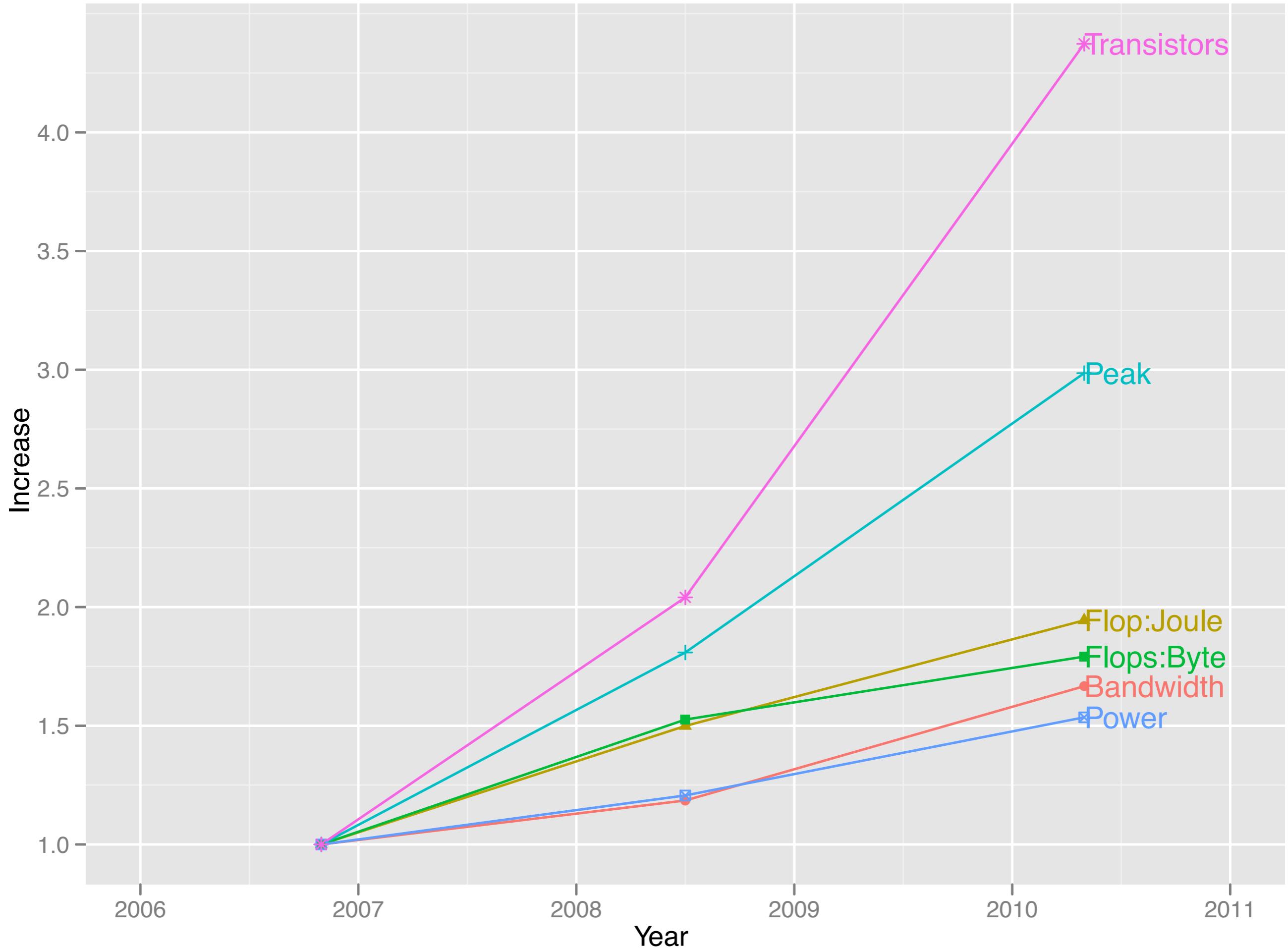
Node











Parallel Sorting (survey)

(Does **not** include Merrill & Grimshaw '10)

